AdvOSS AAA: Architecture, Call flows and implementing emerging business use cases

An AdvOSS White Paper

Latest version of this white paper can always be found at
http://advoss.com/resources/whitepapers/AdvOSS-AAA-workflows.pdf

Introduction

A modern AAA system requires implementing complex service logic for Authentication, Authorization and Accounting tasks. Several operations are performed during the processing of AAA requests that may require interactions with external systems such as databases for retrieval of user credentials and profiles, third party enterprise systems such as CRM, DHCP servers for retrieval of Hardware or IP Addresses, rates and quotas from Rating engines and management of AAA sessions to name a few. Similarly, modern AAA systems may require operating in an asynchronous push mode from time to time where they would need to push commands to their clients asynchronously resulting in changes in subscriber experience.

Sequence of operations for these tasks may also be different in different deployment scenarios. These types of interactions with other systems, and the implementation of service logic to realize new and ever emerging business use cases requires that AAA systems must provide a call flow programmability layer.

The challenge in providing such a programmability layer is that it must be decoupled with the protocol front-end layer i.e. RADIUS or DIAMETER so that the standardized protocols could be implemented and provided on the front-facing part of AAA without concerning with the changes in service logic on the back-end. The programmable service logic must be easy to modify on a running system to avoid long turn-around times in implementing new features in AAA system.

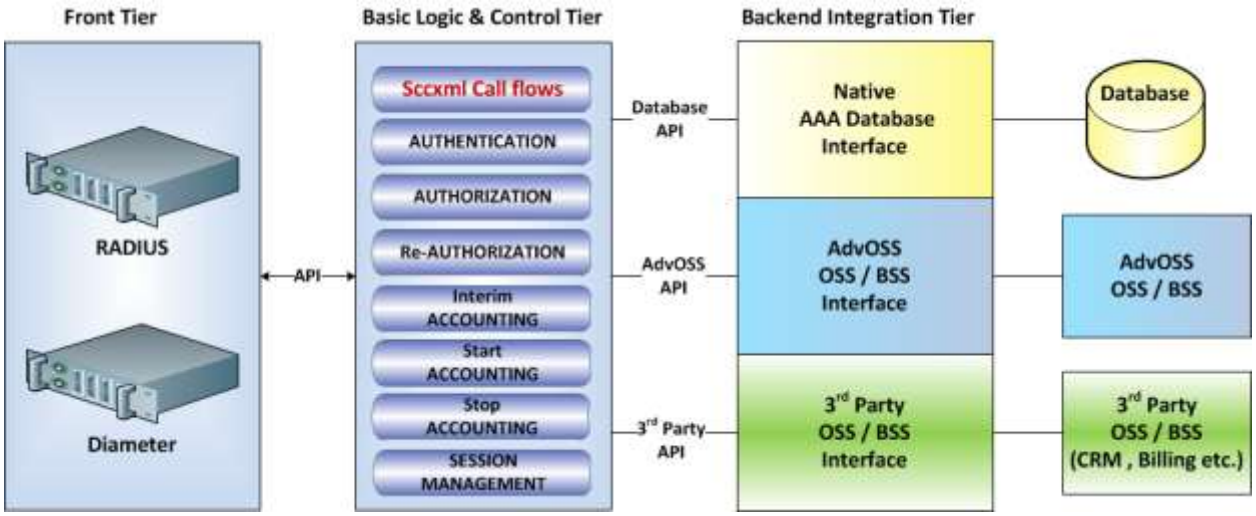
In this paper, we first propose that a three tier Service Oriented Architecture is required in modern AAA systems to meet the requirements of today's competitive CSP market. Then we describe how AdvOSS AAA realizes this architecture. The paper follows up with several business use cases that would be very difficult to realize at a brisk pace in modern CSP deployments without the call flow programmability and a loosely coupled three tiered architecture.

A three tier SOA Architecture

Drawing an analogy with the three tier architecture of the modern web-services, we need a similar three tier architecture for AAA systems. The three tier architecture in web-services in loosely coupled and divided into three tiers of presentation, service logic and persistence layer (databases).

If an AAA system can follow a Service Oriented approach while realizing a similar architecture where each tier exposes its services via some standard APIs to its adjacent tier could make the system much more flexible and manageable for future growth, increase scalability and enable it to accommodate new business use cases at a brisk pace.

The following tiers can be identified in the proposed three-tier architecture of a AAA system:



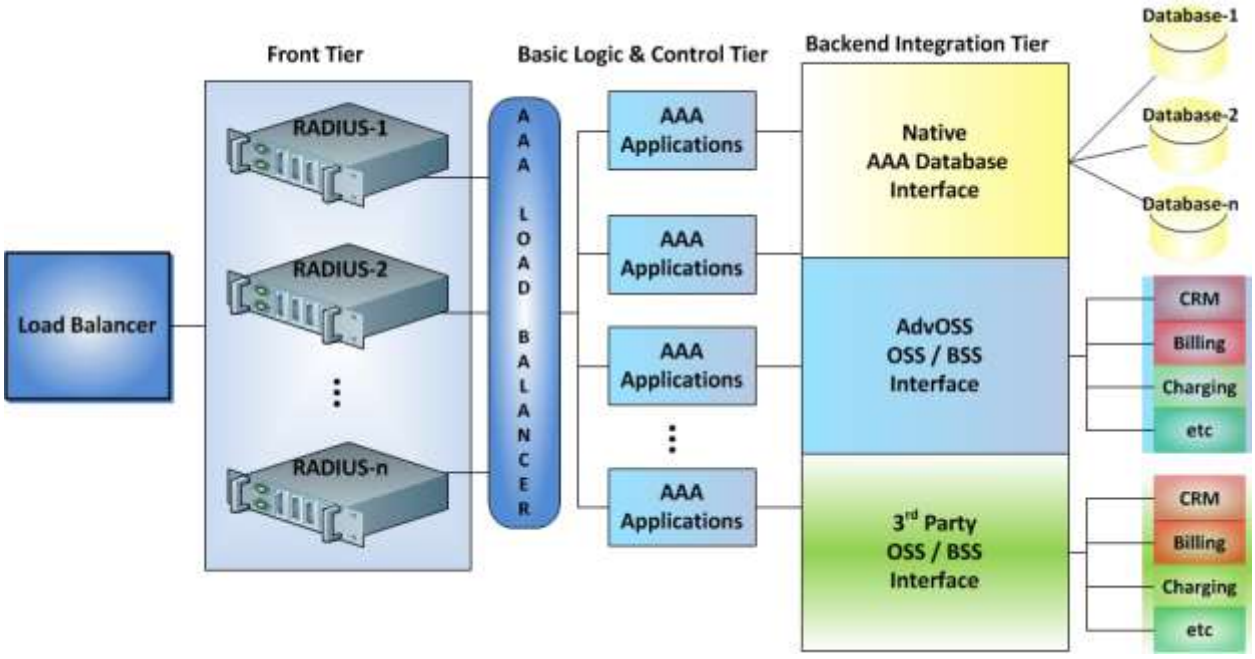
- **Protocol Front-end tier**
  - This is the front tier that interfaces with the clients on a standardized protocol such as RADIUS or DIAMETER. This performs the following tasks:
    - Receives incoming AAA requests from clients
    - Parses and checks the validity of the requests as per protocol standard
    - Uses the API exposed by the middle tier to invoke the particular Call Flow appropriate for the request, passing it the data received in the request.
    - Exposes its own API to the middle tier to receive data from the middle tier and pack it in protocol specific packets and send the responses to the AAA clients.
    - It maintains any protocol specific state such as protocol sessions if required by the protocol standard.
- **Call Flow and Business Logic tier**
  - This tier provides the Call Flows or scripts that implement the business logic and use cases for CSPs. In other words, these are AAA applications that handle all the complexity of information retrieval from back-end system and then conditional or un-conditional evaluation of actions based on the information received in AAA packets from the protocol front-end or the back-end information systems.
  - It integrates on the back-end with two type of systems:
    - Databases native to AAA system for user credentials such as Authentication data, or for session management of AAA sessions.
    - Third party systems for retrieval of information regarding the particular request being processed e.g. with HSS (Home Subscriber Server) to retrieve subscriber's service related profile, rating and charging information, policies applicable to subscriber from a third party policy server etc.
  - It uses APIs exposed by front-end protocol tier to provide the information needed by it to form a AAA response in specific protocol attributes and packets. It also uses the said API to create and send an asynchronous message e.g. Change of

Authorization n RADIUS, to push a particular policy driven action for the subscriber such as redirection to a self-care portal for recharge.

- **Back-end Integration Layer**
  - This tier provides the integration API to Call Flow tier to enable its integration with its own native databases as well as external or even third party systems for information retrieval and storage.
  - This tier provides interfaces for different data management tasks in attached databases and also hides the details of integration with third party systems from the Call Flow tier.

Scalability: Key advantage of Three-Tier Architecture

One of the most important advantages of the three tier architecture described above is that each tier can be scaled individually on its own, without affecting the other tiers as required. For example, we could place a load balancer in front of the protocol front-end tier that distributes traffic across multiple instances of protocol servers. Similarly, protocol servers may choose between multiple instances of applications processes implementing the Call flows, in other words load balance requests among them. In the same way, back-end integration and database tier can be scaled by concentrating on techniques for scaling of databases e.g. sharding etc as load increases on them, while hiding the details from the Call flow tier. This results in a highly flexible and scalable architecture in a truly Pay as You Grow model where different parts of the solution can scale in a loosely coupled environment by adding resources at the right place. In fact this model even creates the possibility of geographical distribution by spreading the three tiers individually in separate, geographically distributed locations.



AdvOSS AAA: A realization of Three-tier Architecture

AdvOSS AAA system is architected in the three tier model as described above. The RADIUS and DIAMETER front-ends are highly efficient protocol processing engines that expose APIs to the Call Flow layer for creation, encoding and decoding of protocol packets. The Call Flow layer is based on an asynchronous event-driven scripting language, while the back-end integration and database layer is based on well abstracted interfaces to the Call flow layer and interacts with external systems and locally attached databases.

In the rest of this paper we will focus on the Call Flow layer in detail and describe its utility and the use cases enabled by it as examples. It should be noted that the use cases presented serve only as examples and more advanced and complex use cases can be easily realized.

SCCXML: AdvOSS Call Flows Scripting Language

AdvOSS AAA exposes an XML and JavaScript combination based programming language that enables implementation of flexible and modifiable AAA applications, customized business logic and call-flows.

This programming language and model used by AdvOSS AAA has been standardized by W3C in **CCXML (Call Control XML)** to write customized call control logic in telephony and multi-media communication environments. The language exposes a simple, finite state machine oriented programming model and supports the embedding of the powerful JavaScript based modules inside XML based scripts.

AdvOSS has enhanced and enriched the core CCXML specifications by adding new primitives, keywords and functionality for AAA and other service management and service delivery use cases, to make it much more useful than the original language in several dimensions. The resulting language has enabled several new use cases not possible in the original language. AdvOSS has named it **Service & Call Control XML (SCCXML).**

SCCXML enables realization of changing business requirements for AAA with ease and in an agile development environment with extremely quick turn-around times after interaction and continuous feedback from the customers and very fast iterations of testing and development. It takes usually a matter of days, not weeks to change the business logic of a particular AAA application on customer demand and put that change in production. This allows for highly flexible, customized and arbitrary business logic, that service providers may leverage to differentiate their offered service offerings and products to their customers.

AdVOSS AAA applications are built on top of AdvOSS Service Delivery Engine (SDE) that provides a session based run-time system that executes SCCXML script sessions in its core execution environment.

AAA use cases enabled by Call Flows

Some of the AAA related service management use case categories enabled by call-flows written in SCCXML include:

- Service Management such as:
  - ChangeOfAuthorization (CoA) sending to different NAS clients in different situations
  - Hot-lining, redirections and alerts
  - Bandwidth throttling
  - On demand and asynchronously disconnecting AAA sessions
  - Provisioning Deep Packet Inspection Engines and other network policy enforcement points e.g. Cisco SCE with updated policy profiles in real-time
  - Changing service profiles for subscribers by retrieving them from HSS.
- Implementing customer specific AAA policy variations while performing Authentication, Authorization and Accounting
- DHCP server interactions including both IP address allocation through DHCP request while acting as a DHCP relay agent, or performing a DHCP lease query to retrieve the Hardware (MAC) address from an IP address while Authentication
- Accessing external or third party web-services via SOAP/XML or HTTP based Restful APIs
- Accessing Databases by supporting both query and stored procedure interfaces.

Following section shows some specific examples of service management business use case scenarios and policies that can be implemented in the above categories:

Authentication based on arbitrary RADIUS attributes

In this use case, the NAS client may be sending authentication credentials in an unconventional RADIUS attribute such as NAS identifier in a Calling Station ID or any other VSA that is not normally used for Authentication. In such cases, the Authentication call-flow attached to a particular NAS can be can be easily customized by merely extracting the Authentication key from a different attribute than the usual username (since all received attributes are passed up to the script, and proceed with authentication accordingly. Minor change in SCXML script is required.

Authentication bypass and learning

CSPs may desire to bypass Authentication on a per NAS basis or globally. This may be required when the environment is considered safe in the sense that clients entering the network are considered reliable e.g. EAP-TLS authentication is configured which is to be followed by username/password authentication and certificates are assumed to be reliable enough, but username/password in the database is not reliable and are mostly stale.

In this case, some or all authentication call-flows attached to a NAS can be modified to proceed with authentication and send Access Accept with default service profile or provisioning values for a network element if applicable. Also, the Authentication script usually stores the credentials received in the RADIUS packets in the authentication logging table, and can insert them in the credentials table also after logging them. This will result in the proper user authentication when the same user enters the network next time with same MAC and username/password.

Missing important attributes in RADIUS packets

Certain NAS clients may send AAA packets with missing important attributes. In this case, most AAA servers would reject packets. However, simple change in AAA scripts can implement arbitrary CSP

policies in such cases. For example, certain default values could be configured for missing attributes. Similarly, NULL values can be populated in logs and UDRs and AAA can be considered successful.

Missing AAA packets for sessions

Sometimes, due to network congestion or other reasons, AAA packets may arrive that do not have corresponding sessions in the AAA database. For example, a Start Accounting may be received that does not have a corresponding Authentication done before. In such cases, a policy can be implemented in the call-flow layer to insert the session dynamically at accounting time, assuming that the authentication has been missed somehow thus ensuring service for subscriber.

Concurrent sessions per subscriber

A CSP policy may dictate to allow one or more concurrent sessions of the same subscriber. Multiple policies are possible and can be implemented by the call-flow layer to query a policy server for concurrency check, or to implement a hard coded policy in the call-flow layer such as logging out a previous session and creating the new one when a new session request comes in I the presence of an un-terminated previous session.

Integration and provisioning of different network elements such as DPI engines

CSPs may have requirements for AAA applications to perform real-time provisioning and update of profiles and other configurations on multiple network elements while AAA is going on. The network elements may include but not limited to DPI engines, policy enforcement points and Application Servers etc. This can be achieved by inserting calls to provisioning layer primitives provided by AdvOSS Service Delivery Engine (SDE) to send commands to any external network elements via AdvOSS provisioning engine to their exposed standardized APIs such as SOAP/XML or Restful HTTP APIs. For example, in case of Cisco SCE, AdvOSS Start Accounting call-flow sends commands to bind IP address received in framedIP address RADIUS attribute to a specific username configured on both AdvOSS AAA and SCE. It can also send a profile name or profile attributes defined on SCE to enforce certain policies for this user session. Similarly, it sends another command to SCE on reception of Stop Accounting to unbind the IP address to the username given at Start Accounting time.

DHCP server interactions in a centralized AAA environment

Several deployment scenarios may require CSPs to abstract out back-end services and expose a centralized AAA environment. These deployments may require AAA to integrate with back-end DHCP server and perform multiple types of DHCP queries from the DHCP service and abstract out such interactions from the NAS clients and authenticating users. Two common use cases frequently deployed and supported by AdvOSS AAA are:

- DHCP lease query
- DHCP address allocation request

The first one i.e. DHCP leas query is used to learn the MAC address from an IP Address sent by NAS client in one of the RADIUS attributes. AAA can learn the actual MAC from DHCP server which results in the

correct MAC retrieved for the client. This MAC could then be used by AAA to authenticate the user device or CPE.

In the second case, a CPE belonging to a CSP may not have direct access to the home DHCP server when in a roaming or a partner's access-network area that is shared by the CSP and partners. In this case, CSPs may want the centralized AAA to interact with home DHCP server, allocate IP address and retrieve other configuration information, and send it in Access Accept to clients using RADIUS attributes. This provides centralization of resource allocation and configuration using RADIUS attributes rather than having all network elements (NAS clients such as ASN gateways and Access Servers etc.) interacting individually with multiple service instances in the core network. They interact with AAA only and AAA integrates with back-end services, and retrieves information as required during AAA procedures. This is made possible only by a call-flow and extensibility layer to provide flexibility and abstraction of resources and substantially reduce time to market for shared access network and other such use cases.

Fact reporting to separate logging systems

AAA systems are required to report all the events occurring in a AAA application or Call Flow by sending events to externally located logging servers. This enables advanced reporting and complex business intelligence and may also cater for Lawful Intercept use cases. This is easily enabled by a scripting layer that implements the relevant AAA Application's Call Flow. If a new event is to be reported to a new third party system, it can be added any time without changing the internal source code of the AAA server and waiting for a new release of the software.

Conclusion

Modern AAA systems require ability to be modified on the fly without too much problems. This is because customer requirements keep changing continuously as new use cases emerge and new network elements are introduced that need to be integrated. AAA systems form the heart of the core network in a CSP environment. They must conform to a Service Oriented, decoupled, tiered architecture to be able to scale and upgrade easily with changes in CSP requirements. Integration with new back-end systems or with new network elements in access networks must be transparent and seamless for them. These integrations should be carried out without disturbing the whole system. Furthermore, programmability and service logic modification is a must for catering to requirements of CSPs that enable differentiation in today's competitive environment. Systems built upon tightly coupled, Silo styled architectures are no longer feasible for surviving in a competitive market. Decoupled systems enable easy scalability, geographical distribution and smooth integration. A CSP must evaluate these traits when evaluating AAA solutions to enable a truly Pay As You Grow model i.e. ability to modify and scale the system as new business needs emerge.