Extending AdvOSS IN

Extending AdvOSS IN

IN (Intelligent Network) is a comprehensive set of products including Account Balance Management Functions, Charging and Rating Engines, Real-Time Credit Control, Integration with AAA servers, Web and IVR based interfaces, Open API for connection to different other third party applications like Voucher Management Systems and Franchisee Management Systems, Integration with Service Managers to provision and de-provision services based on different stimuli and integration with a billing engine to name a few.

An IN is running at the heart of any Service Provider and in today's multi-service offerings with multi-tiered pricing structures, IN is the place where all Service differentiation is realized.

Much of the product differentiation is finally realized at the IN and the IN team is charged with the duty to realize all the requirements coming from the products and business groups. The IN needs to be work flow driven with rule based policies working to realize the offerings by the Service Provider. But in today's environments, this is not all what is needed.

Typical architecture of the IN from most competing products is that of a BlackBox with an API. Different products integrate with the IN and provide it with API calls for the IN to do its function.
IN is usually the slave acting on stimuli about Service Usage and Payments as they come from the network.

But creating a Product Differentiation required by today's Service Providers and Operators, requires the IN to be the master in many cases.

Examples where IN needs to be the master driving events:

- The IN creates the stimuli that drive the provisioning engines to throttle and control network usage as per policies defined.
- IN creates the stimuli to apply fair-go policies on specific users as per their contract.

- IN creates the stimuli to trigger loyalty providing work flows.
- IN creates the stimuli to initiate churn minimizing actions.
- IN creates the stimuli to realize promotions and bonuses on the fly when their conditions are met. IN creates the stimuli to trigger Service Management functions to hotline user's services when they run out of credit
- IN creates the stimuli to suspend customer's service when new billing cycles fire if the customers don't have enough credit
- IN creates the stimuli to charge on file payment methods right in between the usage of a Service should be need arise, so as to continue offering Service without an interruption to a Customer with payment methods on file.
- IN may also keep counts of identity theft patterns and fire events where appropriate
- IN may also send events on fraudulent activity or network compromise
- IN may also be scheduled to send events on time-barred activities like stale times for Customers in active state.
- IN sends stimuli when different counters are topped up.
- IN sends stimuli to send notifications when a Customer's balance falls below a preset threshold.

Hooks:

To serve this purpose, AdvOSS IN provides 'Hooks'; a place where customizable code can be simply dropped in to serve the operator's purpose.

A hook is fired each time an event occurs that matches the place of the hook and executes the code attached to that hook.

Many hooks come with existing and functioning sample code which is open source for the operator to Customize. Other hooks come with empty place holders for the Operator to drop his code in there.

Types of Hooks:

Each hook can be defined amongst many types depending on the purpose the operator wants to serve in there.

Type of Call:

A call to a hook can be of two types:

- Asynchronous
- Synchronous

An Asynchronous call sends a notification event to the called process and returns immediately to the main process running inside the IN. Most Asynchronous events are programmed to call a given IN API as their CallBack function when they complete.

Synchronous calls send a notification event to the called process and then wait for it to complete and send a success or failure message. This keeps the main process running within IN in a waiting loop as well waiting for the external work flow to complete.

In-Transaction and Post-Transaction hooks.

AdvOSS IN provides two types of hooks for many possible events.

In-Transaction hooks are called from within an atomic transaction. They have to be synchronous processes where IN waits for completion of child workflow. In-Transaction hooks would rollback the pending transaction in IN in case the child workflow attached to the hook fails with an error. In-Transaction hooks will only commit the IN transaction after they get success confirmation from the external work flow attached to the hook.

Post-Transaction hooks are executed after the IN successfully commits its atomic transaction. They will leave the master IN transaction committed even if the child workflow attached to the hook fails afterwards.

Real-Time and Polled

Hooks can also be designed to be Real-Time in which case they are called right when the event fires, or they can be programmed so the master hook only pushes the event inside a queue which can then be consumed later by any external program that polls that queue.

Types of Code:

AdvOSS IN support at least two types of code to go behind the hooks.

1. PL/SQL based hooks: These hooks are written in PL/SQL language of the underlying chosen database.
2. External Programming Languages: These hooks are written in any programming language supporting to listen on a socket. AdvOSS IN fires an event onto that external socket for each call to the hook. AdvOSS IN provides sample code for C and Java for the Operator code to listen to the socket and receive the notifications of fired events.

Both the types of code can be made to run Synchronously or Asynchronously.

Partial List of Hooks Provided by AdvOSS IN:

- Receipt of payment
- Real-Time balance getting lower than preset threshold
- Movement of balance for any reason from negative to positive (Negative and Positive defined based on Credit profile attached to Customer)
- Movement of balance for any reason from positive to negative (Again Negative and Positive are relative numbers based on a Customer's profile)
- Firing of new billing cycle
- Different self serve events like change of passwords, change of plans, change of add-ons
- Filling up of a counter associated to a Customer for any reason

Examples of work flow that can be achieved using IN hooks

Example 1: Charging of on-file payment methods

When a real-time Service is being delivered e.g. a voice call is going on, the real-time credit control application is doing reservation of credit from the IN periodically. It is usually configured to reserve credit for one to three minutes at one time. During the delivery of this service, the IN finds out that the balance of the Customer after a new reservation was made by Credit Control application, has fallen below the threshold preset to fire payment processing.

AdvOSS IN will immediately fire an event to the relevant hook in this case "Low Balance Post Transaction" hook.

The event is received over a socket connection immediately by the Java code responsible to process the payment. Event payload already contains the CustomerID and the Balance of the customer at the time the event fired. The work flow attached the hook can then query IN or CRM to find out if a payment method is available on file. If yes, then it can also pull the payment preferences from CRM or IN and try and charge the on-file payment method in real-time. If the payment is successful, the external payment processing system will send a Payment received call to IN and IN will increase the balance of the Customer in real-time. If it all happens before the next credit reservation call, the Customer can continue to enjoy his ongoing call with a mid call on demand top up all orchestrated by IN.

Suggested Code Type: Java
Suggested Call Type: Asynchronous
Suggested Transaction type: Post-Transaction

Example 2: A three plus one promo

Business development realizes that the most likely times for Churn to occur is when a Customer's credit is finished and it is most expensive to retain customers at that time. So they find out that giving customers an extra month of credit if they pay for three months in one go.
Engineering and IT team can realize this by dropping their work flow at the "Payment Receipt Post Transaction" hook. Whenever a payment is received, the hook fires passing the CustomerID and the amount of payment received

and a work flow can calculate if a bonus is due and post it using given IN APIs.
Suggested Code Type: PL/SQL
Suggested Call Type: Asynchronous
Suggest Transaction Type: Post Transaction

Example 3: Fair-usage policies
An operator providing access services offers an unlimited plan for a said price. But the unlimited plan only offers a download of up to 20GB per month and anything beyond 20GB is subject to fair-usage rules.

Fair-usage policies may require the Customer's bandwidth to be throttled to a lower value or may require outright charging of extra volume used.

Given this duty, the Engineering team created a Counter of all such customers that would count their usage in real-time and put its cap amount as 20GB. They would then identify the hook that they need which in this case would be "Counter Fill up Post Transaction". They can drop their workflow at this hook. Whenever any counter is filled up, the hook is fired and passes the Counter ID as well as CustomerID to the workflow.

The work flow can decide and fire an event to the Provisioning Engine to throttle bandwidth for this customer or inform the charging engine to start charging for extra volume.
Suggested Code Type: Java
Suggested Call Type: Asynchronous
Suggested Transaction Type: Post Transaction

Example 4: Reduce a Franchisee's balance upon Credit Transfer into a Customer's account
In the IN, when it is plugged into the Franchisee Sales Network, it is possible to allow the Franchisee's to transfer credit into Customer's accounts through different provided interfaces.
For each such transaction, it is important to be able to call a hook for strict credit control and accounting. For each payment posted through a Franchisee network, it needs to be assured that balances for Franchisee are debited always when the balances for Customers are credited.

To achieve the said functionality, the IN team can use the "Payment Receipt In-Transaction" hook to accomplish this. Whenever a payment is received from a Franchisee, an event is fired to this hook and the hook can do the necessary maths for strict credit control and accounting on the Franchisee side. Just because if the Franchisee didn't have enough credit or an accounting error occurred during posting on Franchisee side, it is important to use In-Transaction version of the hook so that the Customer's credit posting application is rolled back as well.

Suggested Code Type: PL/SQL
Suggested Call Type: Synchronous
Suggested Transaction Type: In-Transaction