ACTIV Content Platform Overview

Contents

1.0 Introduction

ACTIV Financial is a market data technology and content company.

The *ACTIV Content Platform* (ACP) is a state-of-the-art market data feed collection and distribution system catering for global financial markets. The ACP is a high performance, scaleable, ultra-low-latency platform that has been designed for the rapidly increasing data rates being experienced in our industry. The ACP consists of multiple server applications and APIs that can be deployed in a multitude of different

configurations or architectures. This document gives an overview of the different ACP server applications, APIs and basic architectures.

2.0 ACP Server Applications

Among the different possible architectures, the ACP supports a consolidated feed architecture called *ActivFeed* and a direct feed (or DMA) architecture called *ActivFeed Direct*. Fig 1 below depicts a hybrid of the ActivFeed and ActivFeed Direct architectures. This diagram actually manages to contain 10 of the 11 different ACP server applications (along with all 4 of the ACP APIs):

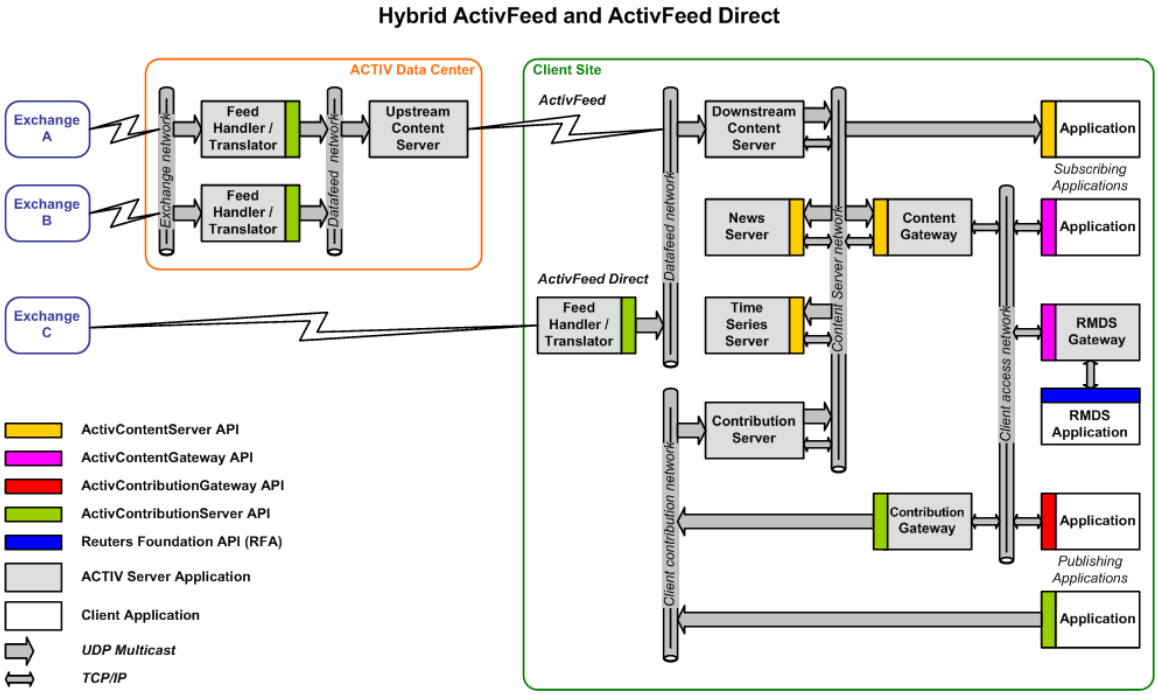

Fig 1: Hybrid ActivFeed / ActivFeed Direct Architecture

Fig 1 depicts Exchanges (in an indigo border), an ACTIV Financial data center (in an orange border) and a client site (in a green border). The ACTIV data center receives exchange data from Exchanges A and B, and generates a consolidated *ActivFeed* stream that is delivered to the client site.

Also depicted in Fig 1 is the delivery of exchange data from Exchange C directly to the client site (i.e. not via an ACTIV data center) and processed directly on the client site. This architecture is called *ActivFeed Direct*.

Several other ACP architectures are possible and these are discussed in section 4.0 below.

The 11 different server applications that constitute the ACP are now discussed:

2.1 Feed Handler

The *Feed Handler* (FH) application is responsible for reliably receiving the particular Exchange feed (or other data source) and passing the contents to the next application tier (the Translator).

2.2 Translator

The *Translator* (T) application receives the raw data from the Feed Handler, translates it into an internal common format and will also typically enrich the data (by calculating value-added fields). The Translator application is usually deployed on the same physical server as the Feed Handler, and the 2 applications together are referred to as a Feed Handler / Translator (or FH/T).

In Fig 1 above, the FH/T applications are present at the ACTIV data center (where they process exchange data from Exchanges A and B) and also at the client site (where they process exchange data from Exchange C). At the ACTIV data center the output from the FH/T feeds into a server application (the Upstream Content Server) that generates the consolidated ActivFeed whereas, at the client site, the FH/T output feeds directly into the Downstream Content Server. This is an important difference for latency reasons as the ActivFeed Direct architecture has 1 fewer application hops than ActivFeed (as well as having 1 fewer network hops).

2.3 Upstream Content Server

The *Upstream Content Server* (UCS) applications are only ever present at an ACTIV data center. They receive multiple enriched, translated message streams from the FH/T applications or Contribution Gateways and generate a smaller number of compressed, higher-bandwidth, consolidated ActivFeed streams. Each ActivFeed stream includes a forward error correcting (FEC) code and diverse routing capability which together ensure reliable WAN delivery. At the client site, the ActivFeed stream is received and processed by the Downstream Content Server.

The UCS also cycles through every record in each UCS database table, filling any spare output bandwidth with full refresh record images. This enables the last-value cache in the Downstream Content Server applications to be seeded as well as enabling them to recover from any feed loss (e.g. due to loss of primary and backup connectivity).

2.4 Downstream Content Server

The *Downstream Content Server* (DCS) applications can receive both ActivFeed data (delivered from an ACTIV data center) and ActivFeed Direct data (delivered from locally-sited FH/T applications).

The DCS builds a complete, persisted, last-value cache of all received records. The DCS also contains a rules engine that is used for generating additional derived fields (e.g. highs, lows, volumes etc) when a record is received. This helps minimize bandwidth usage on ActivFeed as well as helping to simplify Translator applications.

The DCS cache can be queried by the Content Gateway, and updates from the DCS are distributed out to the Content Gateway and several other server applications that can be connected to the DCS (and are discussed below). These optional value added server applications that can database and index DCS content in more sophisticated ways than the DCS.

The DCS also has the ability to delay received data (in a set of delay queues) according to rules published by exchanges. These delay queues feed into a "delayed" last-value cache, and these records (along with streaming updates to them) can also be made available to the Content Gateway.

2.5 Contribution Server

The *Contribution Server* (CS) application has the same functionality as a DCS but receives content from client applications rather than from ACTIV FH/T or UCS applications.

Data can be published into the CS using either the ActivContributionServer API (shown in lime green above) or the ActivContributionGateway API (shown in red above).

2.6 Content Gateway

The *Content Gateway* (CG) is the application that acts as the server for ActivContentGateway API-based applications (shown in pink in Fig 1). It accepts logons from the ActivContentGateway API, authenticates them and accepts requests for data. If the CG doesn't have a data element cached, then it will query the last-value cache in the DCS for the item. The CG enforces a data permissioning model and will only pass on data that an API is permissioned for. It also maintains watchlists for ActivContentGateway API-based applications and, as updates are received by the CG from the DCS, it will forward on as appropriate.

In addition to connecting back to the DCS, the CG also provides access to data that is held in the News Server, the Time Series Server, the Contribution Server and the Symbol Directory.

2.7 Contribution Gateway

The *Contribution Gateway* is a server application that accepts connections from ActivContributionGateway API-based applications. It passes data from these applications through to the CS and enforces a permissioning model that can restrict:

- The client permissions that an ActivContributionGateway API application can use.
- The tables into which an ActivContributionGateway API application can publish.
- The exchange codes that an ActivContributionGateway API application can use.
- The maximum message publishing rate for an ActivContributionGateway API application (not currently implemented).

The Contribution Gateway is usually deployed at a client site but can also be hosted at an ACTIV data center (see Fig 2 below) and allow remote ActivContributionGateway API applications to contribute data into hosted DCS applications or into the consolidated ActivFeed streams.

2.8 Time Series Server

The *Time Series Server* (TSS) processes the output stream from the DCS and the CS. Unlike the DCS, the TSS databases quote and trade data tick-by-tick and also supports bar building. It also contains a historical database of daily values that can go back in time up to 15 years. The CG can connect back to this server and satisfy time-series requests from ActivContentGateway API-based applications.

2.9 News Server

The *News Server* (NS) processes the output stream from the DCS and the CS. The NS can store and index millions of news stories that are received. The NS database supports fast searching (30 millisecond response time average) for stories by date, symbol, category code, headline and story body content. The CG can connect back to the NS and satisfy news story requests from ActivContentGateway API-based applications.

2.10 Symbol Directory

The *Symbol Directory* (SD) is a server application that is usually deployed on the same server as the Content Gateway. The CG connects to the SD and satisfies the GetSymbols ActivContentGateway API method by passing the request through to the SD process.

The GetSymbols ActivContentGateway API method offers functions for searching instrument names and local codes to get matching symbols. There is also an option to filter the matching symbols on entity types.

2.11 RMDS Gateway

The *RMDS Gateway* is a server application that allows ACTIV data to appear as a data source on a Reuters RMDS market data system. Sink applications that are written using the RFA 5.1 or RFA 6.0 can therefore receive ACTIV data.

3.0 ACP APIs

There are 4 different APIs in the ACP, 2 for subscribing to data (the ActivContentGateway API and ActivContentServer API) and 2 for publishing data (the ActivContributionServer API and the ActivContributionGateway API).

All 4 of the ACP APIs (and indeed all of the ACP server applications) are actually built on a 5th lower-level API and runtime called ActivMiddleware. ActivMiddleware contains a platform abstraction layer, a realtime database engine, a suite of messaging components and a component-based application framework and a set of base services for building large-scale distributed systems.

All the APIs are modern, high-performance C++ APIs and have broad support for many different CPUs, operating systems and compilers. An up-to-date platform list is available here: http://www.activfinancial.com/platforms.htm . The ActivContentGateway API and ActivContributionGateway API also have pure Java and pure C# implementations.

Many 3rd-party Independent Software Vendors (ISVs) have used the ACP APIs to build value-added applications. Some of these ISVs are listed here: http://www.activfinancial.com/partners.htm .

3.1 ActivContentGateway API

The ActivContentGateway API (shown in pink in Fig 1) is a programming interface used for querying and subscribing to any data held in the ACP. It connects to the Content Gateway. Its clean, modern, high-performance design enables 30,000 or more records to be snapshot per second. It has been specifically engineered to work well in a WAN (as well as a LAN) environment:

- It operates over TCP and is Network Address Translation (NAT) friendly.
- The wire-protocol minimizes bandwidth by using bandwidth efficient binary formats.
- Advanced server-side filtering (field-level and event-type) further reduces bandwidth.
- Server-side streaming software and hardware compression facilities can be enabled from the client-end, reducing bandwidth further.
- The wire protocol supports compound requests and is not “chatty”, enabling large numbers of records to be retrieved with minimal round-trips.

Detailed programming information about this API can be found in [1].

3.2 ActivContentServer API

The ActivContentServer API (shown in gold in Fig 1) can be used for bulk subscription to the update streams delivered by the DCS or CS. These streams can be very high bandwidth and are delivered over UDP multicast. As no connection is made to the Content Gateway, subscribing applications that require most or all of the data from a DCS or CS output stream can use this API without putting any load on a CG.

3.3 ActivContributionServer API

The ActivContributionServer API (shown in lime green in Fig 1) can be used to contribute bulk data into the CS over UDP multicast. Data is published directly into the CS (without filtering of any type), as such, this API should only be used by trusted applications.

3.4 ActivContributionGateway API

The ActivContributionGateway API (shown in red in Fig 1) can be used to contribute data into the CS via the Contribution Gateway over TCP. As the delivery is TCP, this is suitable for contribution into an ACP over WAN or LAN-based inter-networks.

As described in section 2.7, ActivContributionGateway API-based applications need to login to a Contribution Gateway. As the Contribution Gateway enforces a publishing permissioning model, this API can be used by un-trusted publishing applications.

4.0 ACP Architectures

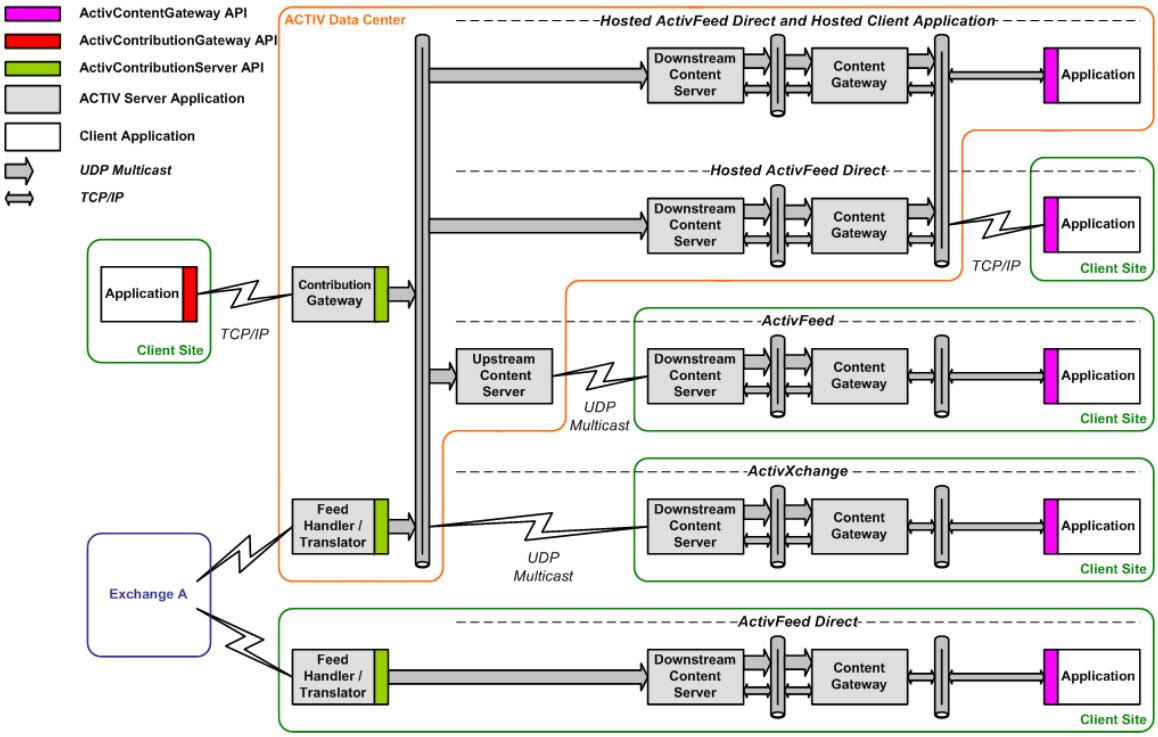

Fig 2: ACP Basic Architectures

Fig 2 above shows the 5 basic architectures for the ACP components.

To support the different architectures, ACTIV maintains multiple redundant ACP infrastructures or *ticker plants* at facilities in New York and Chicago. ACTIV also has several Points Of Presence (POPs) in North America including San Francisco, Toronto and on the floors of the CBOE and AMEX. More POPs are planned. Internationaly, ACTIV has datacenters in Tokyo and Hong Kong with Singapore and London opening shortly.

Not all data sets that are collected at each ACTIV ticker plant are available on all products. Specifically, not all content is delivered over the consolidated ActivFeed although ACTIV supports all feeds in the ActivFeed Direct product. An up-to-date content list is available here: http://activfinancial.com/exchangelist.htm .

4.1 Hosted ActivFeed Direct and Hosted Client Application

In this configuration, all components (server applications and API) are physically located at an ACTIV data center. Typically the client will need some access to their application in our facility (e.g. VPN access over the public Internet or a leased line). This architecture is suitable for low-latency clients who wish to run in an infrastructure-free environment or clients who wish to consume huge volumes of data without having to provision high-bandwidth WAN links.

4.2 Hosted ActivFeed Direct

This is a common architecture for many clients. In this architecture, the only component at the client site is the ActivContentGateway API that is embedded in the client application. This use case has similar benefits to 4.1 i.e. clients don't need to locate servers at their facility. Clients will typically connect their application to the Content Gateway over a leased line (e.g. metro Ethernet) or the public Internet. Note that this can still be a very low-latency solution (metro Ethernet latencies can be substantially under 1 millisecond) and the ActivContentGateway API has been engineered to work exceptionally well in a WAN environment (see section 3.1 for more).

4.3 ActivFeed

This is ACTIV's consolidated feed solution. In this architecture, a packaging device (the UCS) takes the output from the many FH/T outputs and produces a smaller number of highly-compressed, forward-error-corrected UDP multicast streams that have been engineered for efficient and reliable WAN distribution. The architecture is higher latency than other Activ architectures as the UCS's time-sliced compression and FEC-code generation introduce a 40 millisecond delay

4.4 ActivXchange

This architecture delivers the uncompressed FH/T output into a client's DCS and CG environment. This architecture can be useful in a situation where a client is proximate to the FH/T outputs (e.g. cross-connecting to an ACTIV ticker plant at a hosting facility).

4.5 ActivFeed Direct

This is ACTIV's lowest latency ACP architecture (along with Hosted ActivFeed Direct architectures). In this model, data is delivered from the exchange or other source direct to the client facility. Note that hybrid solutions of this along with architecture 4.3 (ActivFeed) are very common and allow a client to take bulk datasets from ACTIV supplemented with direct feeds for datasets that need to be as low latency as possible. This hybrid architecture is depicted in Fig 1.

End-to-end latencies for this model and the hosted variants (i.e. from exchange input through the ActivContentGateway API and into an application) are under 0.4 milliseconds. Latencies for the ActivContentServer API (i.e. bypassing the CG) are under 0.3 milliseconds.

4.6 Other Architectures

As described in section 2.0, hybrid architectures are possible (e.g. hybrids of ActivFeed and ActivFeed Direct are common).

It's also possible to have different architectures at different client sites (e.g. ActivFeed Direct at the primary site with ActivFeed at a backup site are quite common).

Finally, it's also possible to instantiate the different ACP APIs multiple times inside a single application (e.g. it's common to have different datasets available at different sites using different basic architectures and have a single application instantiate the ActivContentGateway API multiple times connecting to different Content Gateways and aggregating the different datasets inside the application).

5.0 ActivFeed Delivery

The consolidated ActivFeed actually consists of 4 different streams. Each stream is actually generated by a fault-tolerant cluster distributed across our 2 main datacenters. This design is depicted in Fig 3 below:

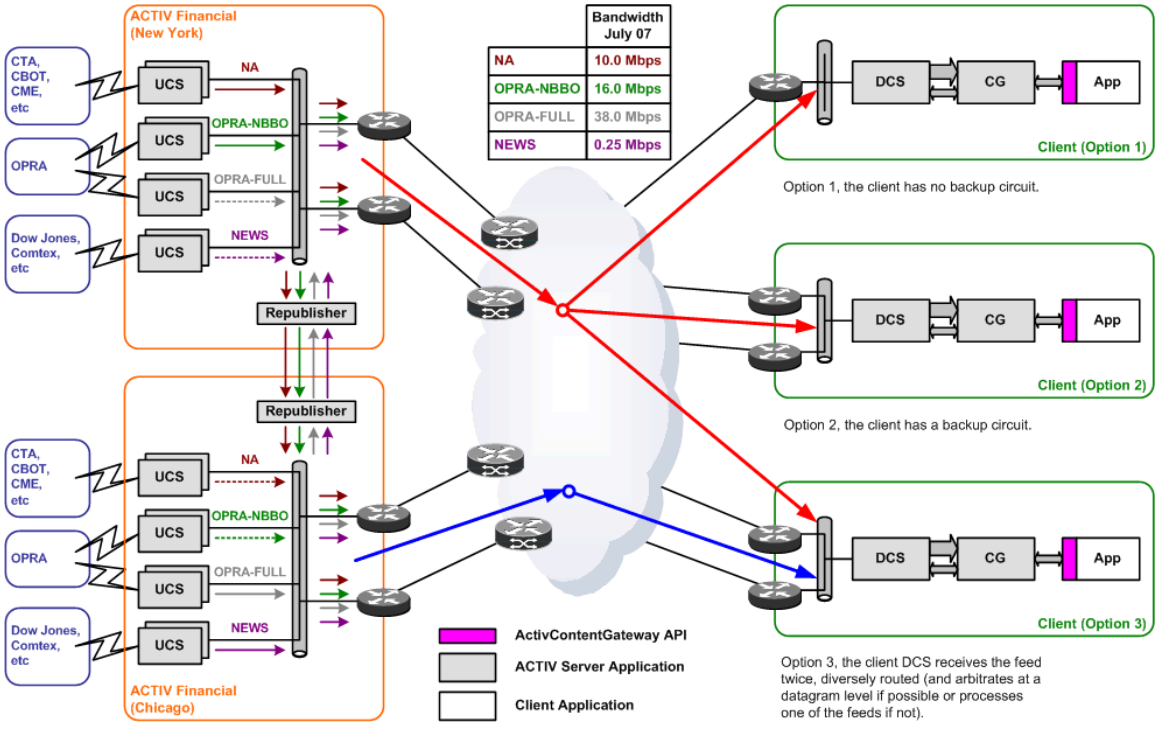

Fig 3: ActivFeed Delivery

The four ActivFeed streams are:

- NA This "North American" stream contains equities, futures, fundamental, forex and administration data. All ActivFeed subscribers must receive this stream.
- OPRA-NBBOThis optional stream contains the NBBO data from OPRA.

- OPRA-FULL This optional stream contains the NBBO data from OPRA and all the regional records as well.
- NEWS This optional stream contains news.

6.0 ActivWorkstationServiceDaemon

The ActivWorkstationServiceDaemon (AWSD) is a Win32 or Win64 windows service or 32-bit / 64-bit Linux daemon that allows multiple applications to run on a single desktop sharing the same permission set. The WS thus acts as a desktop "repeater" application i.e. it uses the C++ ActivContentGateway API to connect to a Content Gateway (CG), and provides a CG like service to local applications (running on the same machine or "desktop") that use the ActivContentGateway API. The Windows AWSD actually comes packaged with the ActivWorkstation (AWS) - although we can (and do) ship this independently from the AWS. The Windows version of the AWSD is a proper Windows service i.e. can be stopped and started through the Service Control Manager (SCM), but also supports an ActivMiddleware telnet-based user interface.

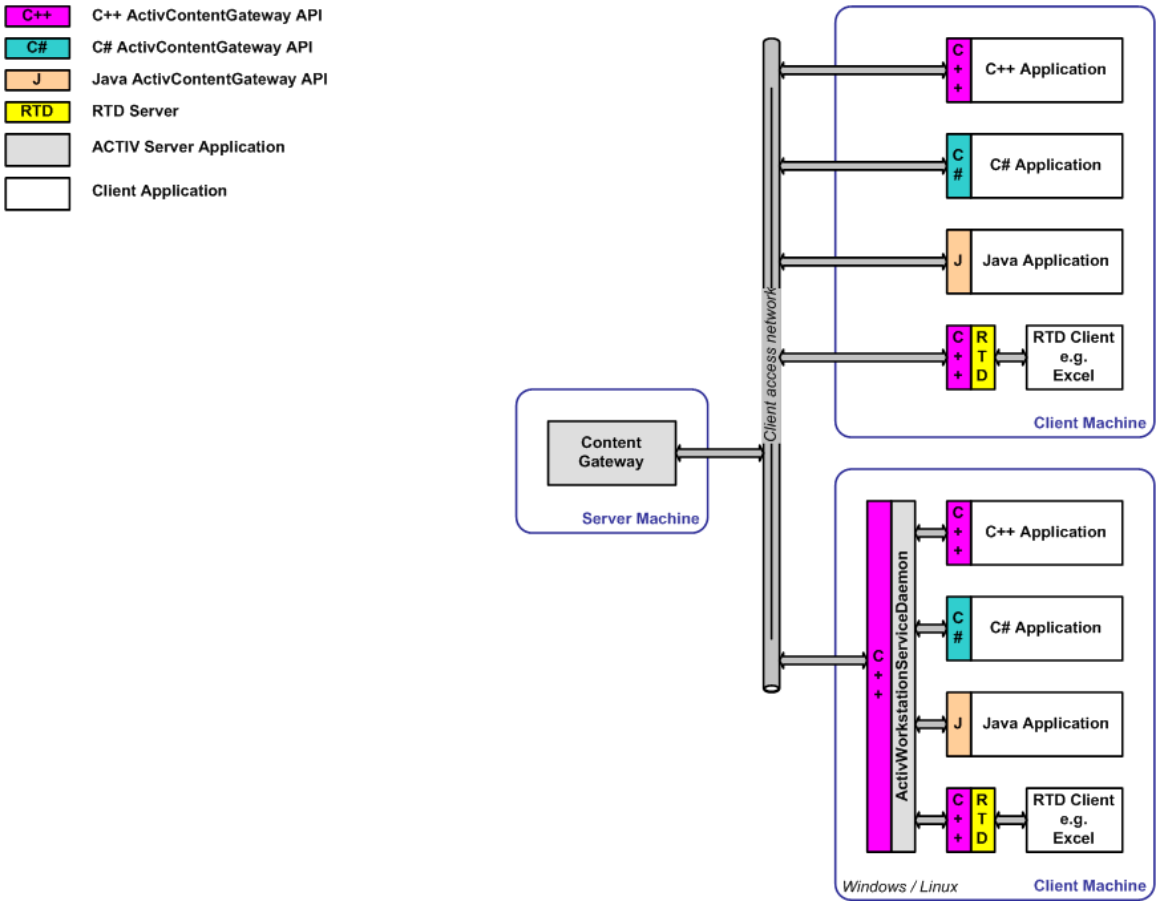

Fig 4: ActivWorkstationServiceDaemon

Fig 4 above depicts a CG running on a server along with 2 client machines. One of the client machines has several ActivContentGateway API-based applications that connect directly to the CG. The other client machine has a AWSD running on it. Connected to the AWSD are 4 applications: a C++ application, a C# application, a Java application and an RTD client application (e.g. Excel):

The basic function of the AWSD is to allow multiple applications running on the same server to share the same login id. Note the AWSD only allows local ("on box") applications to connect i.e. it won't allow an ActivContentGateway API-based application on another machine to connect to it. It does this by only listening on the local TCP loopback addresses (both IPv4 and IPv6) for ActivContentGateway API based applications.

Should you have just a single ActivContentGateway API-based application running on a desktop, then it doesn't make sense to connect it to the AWSD. However, when you have 2 or more applications running on a single desktop, then it can make sense to use the AWSD. Firstly it means that we only need to give out one login id (that gets shared by both applications) and this can be an exchange fee cost saving to the user. Secondly, it means that there is potentially less work for a CG to do i.e. it only needs to send an update out once to an AWSD and the AWSD will fan the data out to any of its connected applications.

7.0 References

[1] ActivContentGatewayApi.pdf. This document is in the docs/ActivContentPlatform/ActivFeedSdk directory of the ActivFeed SDK.