CA Gen Integration Solutions

White Paper

Christian Kersters
Senior Services Architect
CA Technologies

V3.1, August 2017

Executive summary

In the last few years, the economy of the world has changed dramatically, and software is now rewriting most, if not all businesses.

To succeed (not to say to survive) today, organizations must move to a Modern Software Factory.

Discussions with many members of the CA Gen community around the world shows that not all of them realize that CA Gen assets are future-proof, ready to take place at the core of the Modern Software Factory, responding with agility to the most demanding business requests, as they have done for decades.

Thanks to its unique architecture, sound design principles, openness and rich ecosystem of complementary solutions and partners, CA Gen puts customer organizations in an ideal position to fuel the agility their business needs with minimal efforts and risks.

This document presents the solutions available in and around CA Gen to leverage the invaluable repository of business logic at hand and make them seamlessly evolve to best profit from the new application landscape.

Key to this evolution are web services, the *de facto* standard for loose integration, which CA Gen can produce or consume. This white paper describes all the possibilities provided by the product and its ecosystem, for both SOAP and REST.

If not enough, and for a comprehensive view on the possibilities of integration of CA Gen, the document also describes more traditional ways of interacting with external systems through lower-level APIs, in- or cross-process.

Contact CA Technologies if you need information or assistance in moving your CA Gen application portfolio to the core or your new software processes, or implementing any of the techniques described in the document.

Contents

Introduction

In the last few years, the economy of the world has changed dramatically, and software is now rewriting most, if not all businesses.

To succeed (not to say to survive) today, organizations must move to a Modern Software Factory.

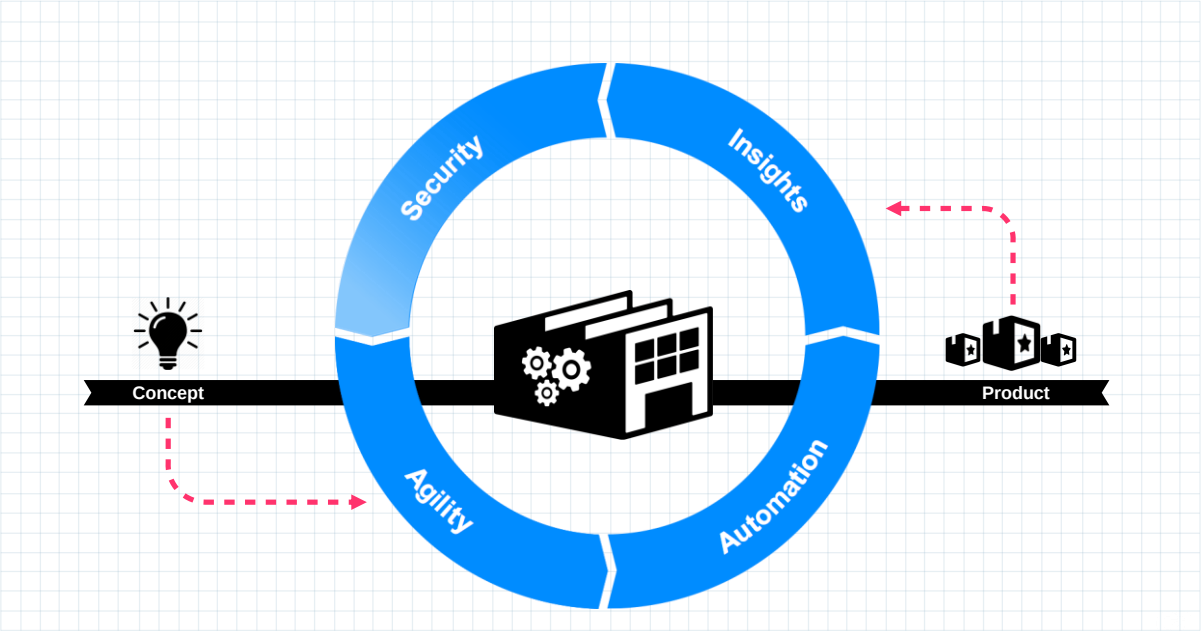

Figure 1 The Modern Software Factory – A blueprint for success

This document only focuses on a part of the factory: **Agility**. To achieve agility, organizations have adopted the Agile Development methodology, moved their focus to mobile platforms and (micro-)services and jumped into the DevOps wagon.

Due to lack of knowledge or tools, however, many companies have missed one cornerstone of the software revolution: ***reuse by composition*** (see figure 2, below). So many of the software assets an organization has invested in over decades, built by so many talented developers, are still valid in the new economy! Ignoring or rebuilding them would be a fatal error, displaying the vain agility of La Fontaine's hare.

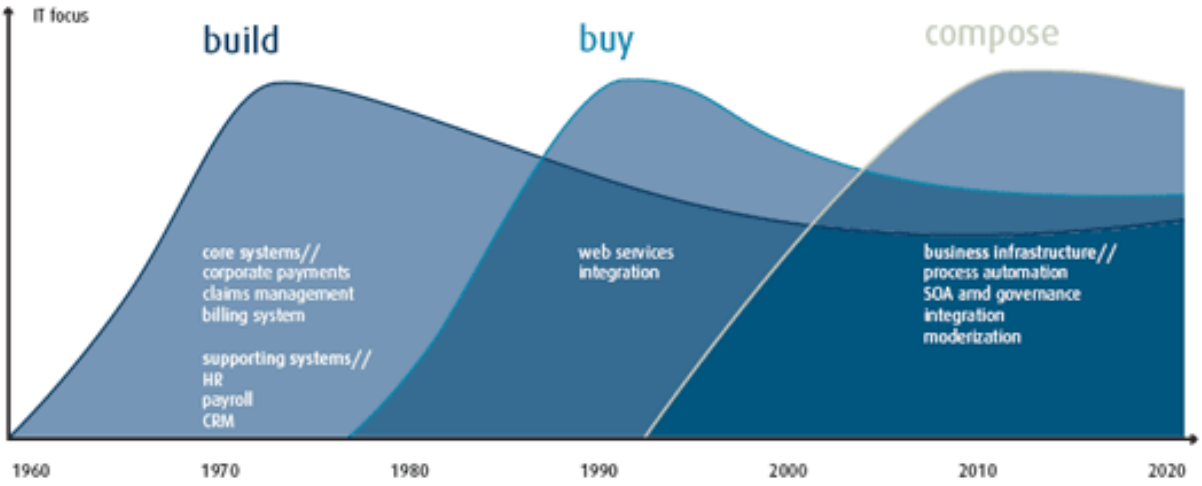

Figure 2 Evolution of the software industry[i]

Thanks to its unique architecture, sound design principles, openness and rich ecosystem of complementary solutions and partners, CA Gen puts you in an ideal position to fuel the agility your business needs with minimal efforts and risks.

Through this document, you will see what solutions are available to you to leverage the invaluable repository of business rules you have implemented over the many man-years of development of applications with CA Gen and how to make them evolve to best profit from the new app landscape.

This white paper will first focus on SOAP and REST Web Services (WS), as today's *de facto* standards for loose integration, and present, compare and contrast the different WS-based integration techniques available in CA Gen.

Reuse is, however, not limited to publication / consumption of Web Services, and CA Gen also provides a rich set of techniques for tighter integration, which together with its model-based approach and the multiple deployment platforms it supports, make it an investment of choice wherever future-proofing matters.

To help you succeed in this revolution, CA Services brings you decades of expertise of software development and integration in demanding, multi-faceted, ever changing environments.

Integration overview

When talking of interactions between applications, we must distinguish 2 very different kinds: Peer-to-Peer (or P2P in short), and Client/Server (C/S).

1. **Client / Server interactions**: in this scenario, tasks are shared between a provider of a resource or service (the Server) and requesters of such resource or service (the Client). That means that Clients initiate requests and Servers wait for incoming requests to process.
   Clients and servers can be tightly coupled, running in the same process context, or loosely coupled, with inter-process communications within one system or over a network
2. **Peer-to-Peer interactions**: here, tasks or workloads are shared between processes. Those processes are peers, equally privileged, equipotent participants in the application. (A typical example of P2P is a chat application).

In the case of business applications, by far the most common mechanism is C/S communications, although they can be bi-directional, meaning that one application sometimes acts as a client to another, and sometimes serves requests from the other application. In the specific case of CA Gen, server transactions are stateless, which makes them ideally suited for use in multiple contexts.

CA Technologies has long recognized the need to open their CA Gen applications to let customers use the solutions that best fit their environment, requirements and constraints. So, in both cases, there are multiple ways in which CA Gen can integrate with 3rd-party products, and growing.

Those solutions can be examined according to a number of axes:

- CA Gen application acting as a Client vs. Server
- Tight vs. loose coupling
- Open vs. proprietary interface
- Product-based vs. Field Pack vs. Partner's solution
- Pre-requisite CA Gen version
- Supported platform / language
- Point-to-point or bus-based

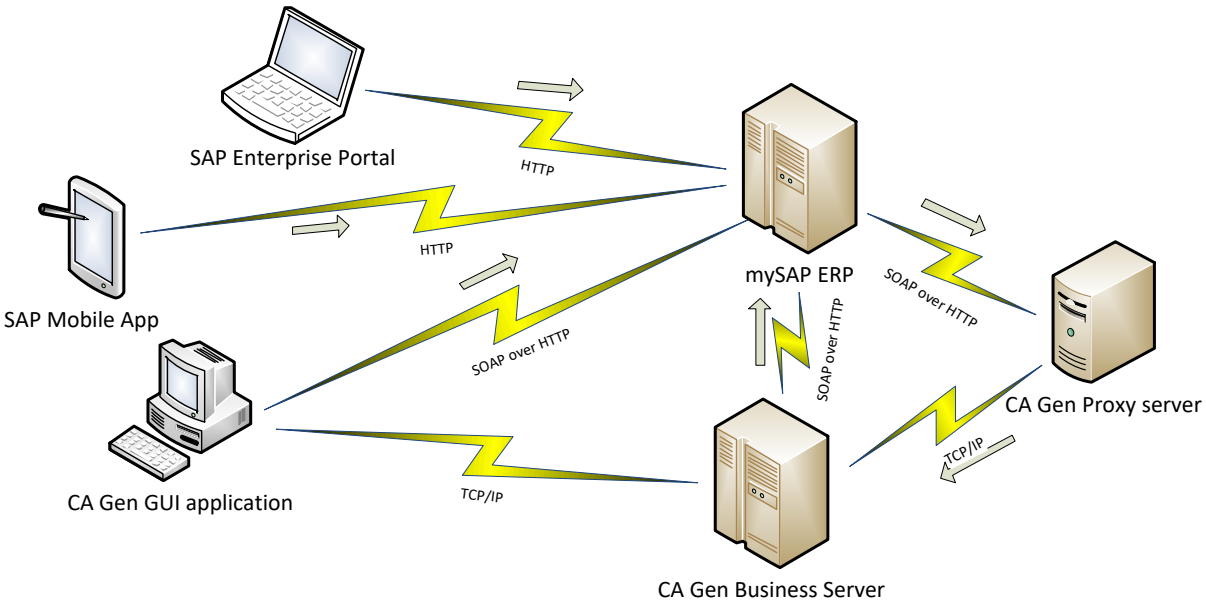

Figure 3 Example of complex point-to-point C/S interactions with CA Gen application

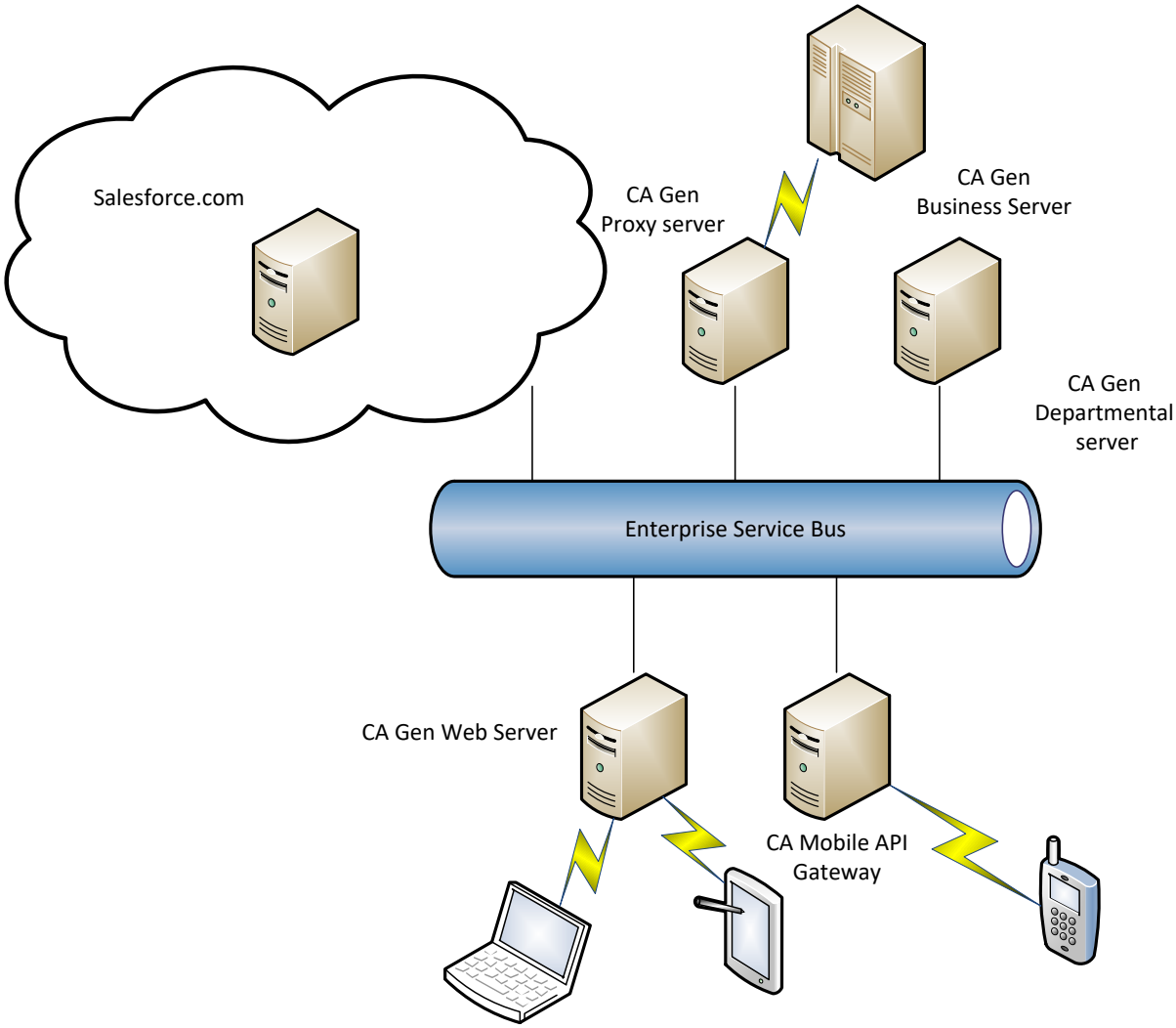

Figure 4 Example of ESB-based integration

CA Gen and Microservices

Microservices

"A *microservice* is an independently deployable component of bounded scope that supports interoperability through message-based communication".

"Microservice applications share some important characteristics:

- Small in size
- Messaging enabled
- Bounded by contexts
- Autonomously developed
- Independently deployable
- Decentralized
- Built and released with automated processes"[1]

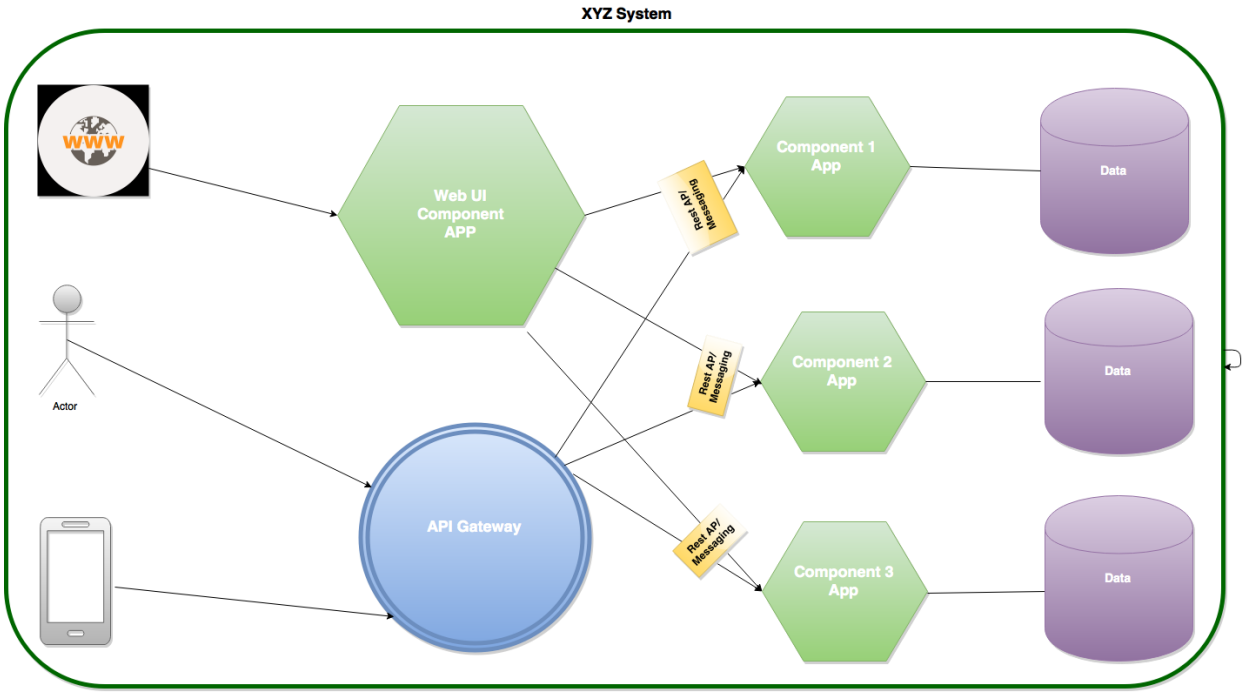

Figure 5 Microservices architecture[2]

Component-Based Development

CA Technologies has promoted the use of Components since 1996, and many customers, since then, have adopted a CBD approach, to some depth, to increase the flexibility of their developments.

[1] Irakli Nadareishvili, Ronnie Mitra, Matt McLarty & Mike Amundsen: Microservice Architecture, O'Reilly
[2] Microservices Architecture : What, When and How, http://www.tipstr.in/microservices-architecture

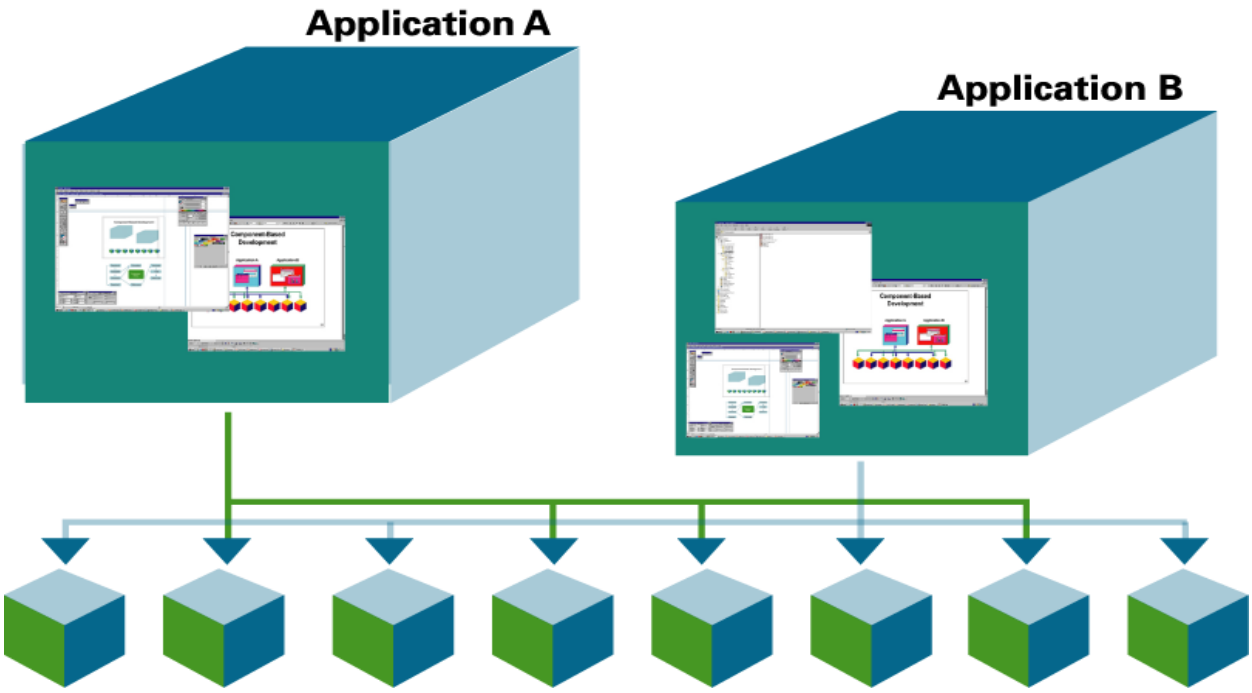

Figure 6 Component-Based Development

A component is a software building block used to build applications or other larger components.[3]

The main characteristics of components are:

- A component is a software building block that can be used to construct applications or larger-grained components and is made available as an independently delivered software package
- A component has three facets:
  - Component Specification – the definition of component behavior
  - Component Implementation – the internal design and code that realizes the specification
  - Component Executable – a set of modules that can be executed to provide the specified functionality
- A component is encapsulated. The implementation can be changed without impacting the software that consumes the component
- The functionality of a component is made available through one or more programmable interfaces (groups of related operations)
- Each operation of a component has the following features:
  - It has an operation specification, which defines its behavior and how that behavior must be invoked
  - It belongs to an interface
  - It is a success unit
- A component is replaceable by another component that supports at least the same interfaces
- A component may have a dependency upon other components. This usually means it invokes operations of those components
- A component implementation may use a data store to make its data persistent
- A data store of a component may not be directly manipulated by any other component.

CA Gen Components and Microservices

If we compare those definitions, it is striking how those concepts are close to each other. The addition of a Web Services messaging interface (described later) to CA Gen components easily transforms them into a Microservices Architecture.

[3] This definition and following characteristics are extracted from AllFusion Gen Components Standard 3.1.1 guide, available as part of the AllFusion Gen 7.6 documentation set

Refer to the Solutions Design chapter in this document for examples of approaches to microservices with CA Gen components.

Of course, this would be the ideal world, and the reality of your applications can require some efforts, because:

- Components can provide a restricted set of functionalities, the rest being directly accessed at implementation level. Customers develop and use a limited number of peripheral components, or expose limited parts of big applications as components, while keeping the biggest part uncomponentized
- Components can be invoked at sub-transactional level (action blocks), implying the need for a cross-component transaction manager (usually provided by the DBMS accessing a DB shared among the components
- As there is no enforcement at CA Gen tools level, there might be some breaches to the encapsulation of components, introduced, for instance, to:
    - Make the development or change process easier/faster
    - Manage cross-components referential integrity
    - ...

Whatever your situation, CA Services and partners are there to help you move your CA Gen developments to a Microservices Architecture at a fraction of the cost induced by redeveloping your business logic, by sound guidance and use of advanced tools, inherited from decades of assistance to demanding customers throughout the world.

CA Gen and Web Services

Web services provide a standard means of interoperating between different software applications, running on a variety of platforms and/or frameworks.

This makes them a key component of any modern development framework or COTS application.

Today, web services are mostly built using one of 2 main standards:

- ***SOAP-based web services***, using XML as payload language, mostly used in business-to-business applications, executing coarse-grained interactions
- ***REST-based web services***, mainly using JSON, but also XML, as payload languages, providing fine-grained interactions and favored for mobile applications.

Having long recognized the importance of Web Services, CA Technologies and partners provide solutions for CA Gen to interact with / provide both SOAP and REST Web Services.

As briefly described later, CA Gen web services can be integrated into other CA products, to build comprehensive Web Services development and runtime environments.

Provision

SOAP-based Web Services

Native support

Natively, CA Gen has provided SOAP Web Services for years, accessible through HTTP(S):

- Axis Web Services, through Java Proxies
- .Net-based Web Services, through .Net Proxies
- EJB-based Web Services, with optional XSL transformation
- Web Services middleware to EJB- or Transaction Enabler-based servers.

Support through partner products

- Canam Software also provide abilities to provide SOAP Web Services capabilities in CA Gen, through their SOA Composer product (http://www.canamsoftware.com/Products/CAGenSolutions/SOAComposerforCAGen/Overview/tabid/257/Default.aspx)
- Response Systems' Web Services GENius provides CICS SOAP Web Services for CA Gen Action Blocks or procedures (http://www.response-systems.com/solutions/genius/).

Support of additional requirements through CA API Gateway

As an option, the interfaces of those web services can be enriched, through publication in the CA API Gateway, and support additional requirements like:

- Security,
- Simple orchestrations,
- Further XSL transformations,
- Caching,
- ...

NOTE ON SOFTWARE-ORIENTED ARCHITECTURES

Due to the strong link between interfaces, CA Gen procedures (or action blocks), by default, provide sets of SOAP Web Services that don't qualify as a Services-Oriented Architecture.

Such an architecture is most closely achieved, in CA Gen, when Web Services are created around CBD components, thanks to the clean data model and operations published at the specification level.

No solution, however, will generate a WSDL referring to any consistent data model.

Alternatively, ***CA API Gateway*** can be used to implement a coherent structure on top of an existing set of Web Services and publish a (manually defined) WSDL, making it a consistent whole.

REST-based Web Services

Support through CA API Gateway

The CA API Gateway has the possibility to publish REST Web Services and convert them to SOAP invocations, including conversion between JSON and XML. This is currently the only CA product-based solution available to access CA Gen business servers using REST.

Support through CA Services Field Pack

The ***Web API Designer for CA Gen*** , developed by CA Services, lets easily build state-of-the-art ROA architectures with REST interfaces and support of XML or JSON messaging to CA Gen business assets, through a wizard-based approach, for deployment in Java environments. (CICS- or .Net-based REST Web Services are candidates for implementation, depending on demand).

Support through partner products

Response Systems' Web Services GENius provides CICS REST Web Services for CA Gen Action Blocks or procedures (http://www.response-systems.com/solutions/genius/).

NOTE ON RESOURCE-ORIENTED ARCHITECTURES

Thanks to its unique approach and flexible view matching principles, the ***Web API Designer for CA Gen*** Field Pack provides access to CA Gen server logic using state-of-the-art Resource-Oriented Architectures.

Any other approach with a strong dependency on CA Gen logic interfaces can only provide sets of REST Web Services that cannot qualify as a Resource-Oriented Architecture.

Some steps can be taken towards such an architecture, in CA Gen, when Web Services are created around CBD components, thanks to the clean data model and operations published at the specification level.

Except for the Web API Designer for CA Gen, no solution can generate an API specification (RAML, SWAGGER, WADL, ...) referring to any consistent data model.

Alternatively, ***CA API Gateway*** can be used to implement a coherent structure on top of an existing set of Web Services and publish a (manually defined) specification, making it a consistent whole.

Consumption

SOAP-based Web Services

Native support

- CA Gen "Web Services middleware": it's possible to consume web services from CA Gen clients or proxies to CA Gen servers through SOAP. This feature is however limited to Transaction Enabler and EJBs
- CA Gen also has the possibility to consume SOAP Web Services through HTTP from C, Java, or C# code (in all types of applications, client, server, online, batch)[4].
  Consumption with SSL handshake is on the roadmap for C# and Java, while SSL authentication is at idea stage.

Support through partner products

Consumption of SOAP Web Services is also provided by Canam Software SOA Composer (http://www.canamsoftware.com/Products/CAGenSolutions/SOAComposerforCAGen/Overview/tabid/257/Default.aspx).

REST-based Web Services

Consumption of REST Web Services has been on the CA Gen roadmap for a while and, once it is delivered, CA Gen customers will be able to do more integrations to open their applications to the interconnected business infrastructure.

Complementary CA products

Development environment

CA Application Test

CA Application Test leverage APIs and Web services, allowing you to:

- Rapidly design and debug automated test sets from the moment you've defined your web service interfaces (from WSDL, WADL, ...)
- Access your web services wherever they are deployed (JEE Application Server, ESBs, ...)
- Execute automated unit, functional, regression, integration, load and performance tests.

CA Service Virtualization

CA Service Virtualization simulates constrained or unavailable systems across the software development lifecycle, allowing developers, testers and performance teams to work in parallel independently, for faster delivery and higher application quality and reliability.

CA API Gateway

At development level, CA API Gateway enriches your CA Gen-based web services, increasing reusability and flexibility, through:

- Protocol transformation, from SOAP to REST and XML to JSON and back
- Customizable API composition and virtualization, creating APIs that contain subsets/supersets of your web services functionalities.

CA API Developer Portal

CA API Developer Portal is designed to help you gain maximum value from APIs by establishing a complete, secure platform for apps developers onboarding, engagement and management.

[4] *Consumption of SOAP Web Services from Cobol is currently at idea stage for future implementation*

Thanks to its features, you can get new revenue streams from existing IT investments and expand your market reach around the world.

Production environment

CA API Gateway

In your production environment, the CA API Gateway provides you with:

- Integrated Web, mobile and XML firewalling
- Advanced SAML, OAuth and XACML identity and access capabilities
- Military-grade security certifications
- Control, prioritizing traffic to help ensure APIs remain available and responsive.

CA Application Performance Management

The CA Application Performance Management (CA APM) solution ensures the performance and availability of business-critical applications as well as the end-user experience of customers that access your online services.

CA APM enables organizations to proactively identify and prioritize problems based on business impact and help resolve problems across today's highly complex application environments before they affect users. Built-in Application Behavior Analytics scan the rich set of metrics collected by CA APM looking for anomalous behavior that signals a problem and alerts you to take action.

Response Systems' APMConnect

For customers already leveraging the strength of CA Technologies' CA APM products, APMConnect for CA Gen enables organizations utilizing CA Gen, to get detailed insight inside their executing CA Gen applications on the mainframe. This new insight enables CA Gen customers to identify application hotspots, and deliver valuable cost savings in large and complex applications (http://www.response-systems.com/solutions/apm/).

Other integration solutions

External access to CA Gen business logic

Native support

Proxies

Natively, CA Gen provides external access to CA Gen business logic, by one of the Proxy products:

- COM proxy, for access from Windows development environments supporting COM objects
- Java proxies, for access from Java native code
- .Net proxies, for Windows .Net environments
- C proxies, for access from Windows or Unix platforms.

All proxies work on the same principles: the custom external code invokes remote server procedures through the proxy and the associated CA Gen runtime and middleware.

Native calls

CA Gen EJB and .Net servers can be accessed directly, respectively from Java and .Net languages, through Java Remote Method Invocation or .Net Remoting.

COM Interface to CA Gen windows

All CA Gen windows / dialog boxes are accessible through a COM interface, which makes them usable in the context of external Windows applications.

Field Pack support

Action Block Proxies have been developed by CA Services, with same interface as native proxies, but able to call CA Gen blocks of logic (action blocks or procedure steps) inline, rather than remotely, with following advantages:

- Reuse of logic at finer grain level[5]
- Invocation of CA Gen procedure steps, generated in Java or C#, in a batch / online environment.

CA Gen access to external business logic

External business logic can be accessed, inline, or Client-Server based using externally available libraries or 3rd-party product features, different native ways.

External action blocks

External action blocks have been the traditional, and for long only, way of accessing external logic / systems from CA Gen. External logic, written in the language of or compatible to the target platform language, can be accessed in any CA Gen logic.

[5] Decision to externalize access to action blocks through the Action Block Proxies is however, of course, subject to the requirement that such action block keeps the data consistent with respect to all business rules, or that the maintenance of the consistency is externalized to the calling logic.

CA Gen user-defined functions

It is possible to define custom functions, for use in CA Gen logic. They have some limitations compared to External action blocks, require some specific tooling for definition, but provide more flexibility in the view matching.

Inline code

It's also possible to write native code directly in the action diagram, with the advantage that such code is saved as part of the CA Gen model. That more recent feature is available in all supported environments.

Custom solutions

Customer solutions

To provide a comprehensive overview of the available solutions, it is necessary to mention here that several customers have developed their own integration solutions, at procedure step or action block level, mostly in mainframe environments.

Such solutions have their own merits, but are rooted in non-published interfaces to CA Gen generated and runtime code. It's the responsibility of those customers to take the risk of future incompatibilities with the product and provide the efforts to maintain those custom interfaces.

CA Services - Optimization Services solutions

Finally, would the solutions presented above not fit 100% with customer requirements, one last possibility exists.

Over the years, CA Services has developed expertise in external interfaces and how to effectively and efficiently extend current solutions with automation.

Advantages of solutions designed and developed by CA Services compared to customer solutions are that:

- They are built upon decades of in depth knowledge of CA Gen product, context of use and usage
- Whereas customer solutions always run the risk of loss of knowledge, through employees turnaround, the knowledge of the solution and its supporting technology stays within the CA Services team
- The supporting technology gives an unbeatable efficiency, reducing development times by an order of magnitude at least
- The solutions are provided with optional full support, including:
    - Upgrades
    - Bug fixes

Solutions Design

Introduction

The diagrams below describe typical/simplified interactions between Web Services consumers and pieces of business logic.

In all those diagrams, the colors are keyed per artifact type, as reported on the right.

Also, the descriptions will focus on new code, taking the business code as a given.

CA Gen Axis -based SOAP Web Services

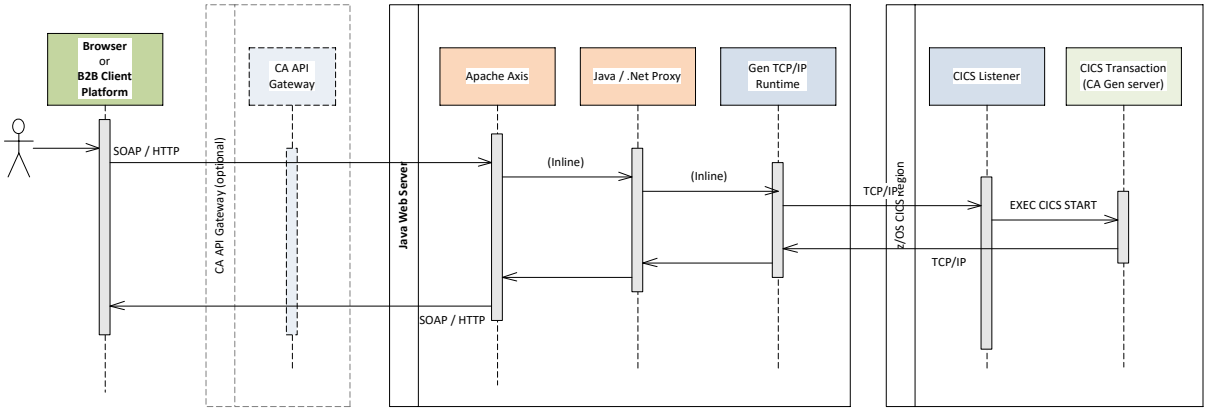

Figure 4 Typical Axis Web Service flow

Creation

To create SOAP Web Services suitable for deployment to the Apache Axis platform, the following approach is taken:

1. In the Developer toolset, the target procedure step is selected, and Java/XML Proxy code is generated for it
2. That code is then "built" (or compiled and packaged) using the CA Gen Build Tool
3. The Web Services Wizard plug-in is then started and the Web Services code and WSDL generated, compiled and packaged
4. The web service and the Java proxy are deployed to the CA Gen Proxy Server (Java Web Server with Apache Axis) and CA Gen runtime configuration is adapted to direct requests to the corresponding transaction.

Processing

1. An HTTP(S) request, with a SOAP payload, is sent to Apache Axis, by the WS consumer
2. Axis calls the WS code, which passes the body of the request to the Java/XML proxy
3. The proxy converts the XML to the required transport format and, through the CA Gen Java runtime, sends it to the CA Gen server transaction, through one of the supported middlewares (TCP/IP in this case)
4. The CA Gen server transaction processes the input and sends a message back, through the same channel.

Options

Flexibility can be added at both ends of the interaction, by:

- Creating a so-called ***façade*** procedure step[6] (as an EJB, for instance) to the server transaction, with as objective to:
    - Provide views of higher level than the implementation-level information sometimes used in tight coupling,
    - Do some pre- and post-processing, to fulfil specific Web Services requirements
    - Orchestrate multiple calls to CA Gen servers
    - …
- Publishing the Web Service to the CA API Gateway, as mentioned before
- Customizing the generators.

CA Gen .Net SOAP Web Services

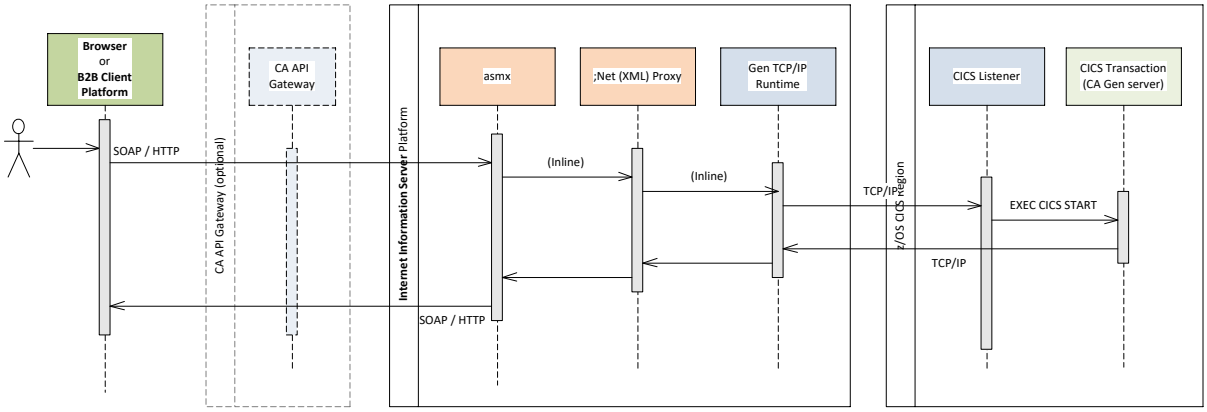

Figure 7 Typical .Net Web Service flow

Creation

To create SOAP Web Services suitable for deployment to the IIS platform, the following approach is taken:

1. In the Developer toolset, the target procedure step is selected, and .Net/XML Proxy code is generated for it
2. That code is then "built" (or compiled and packaged) using the CA Gen Build Tool
3. The web service (asmx) and .Net Proxy are deployed to the CA Gen Proxy Server (Internet Information Server) and CA Gen runtime configuration is adapted to direct requests to the corresponding transaction.

Processing

1. An HTTP(S) request, with a SOAP payload, is sent to IIS, by the WS consumer
2. IIS passes the body of the request to the .Net/XML proxy
3. The proxy converts the XML to the required transport format and, through the CA Gen .Net runtime, sends it to the CA Gen server transaction, through one of the supported middlewares (TCP/IP in this case)
4. The CA Gen server transaction processes the input and sends a message back, through the same channel.

Options

Flexibility can be added at both ends of the interaction, by:

- Creating a so-called ***façade*** procedure step[6] (as an EJB, for instance) to the server transaction, with as objective to:
    - Provide views of higher level than the implementation-level information sometimes used in tight coupling,
    - Do some pre- and post-processing, to fulfil specific Web Services requirements
    - Orchestrate multiple calls to CA Gen servers

[6] CA Technologies recommends this approach, as it gives much more flexible and resilient Web Services

- ...

- Publishing the Web Service to the CA API Gateway, as mentioned before.

CA Gen EJB -based SOAP Web Services

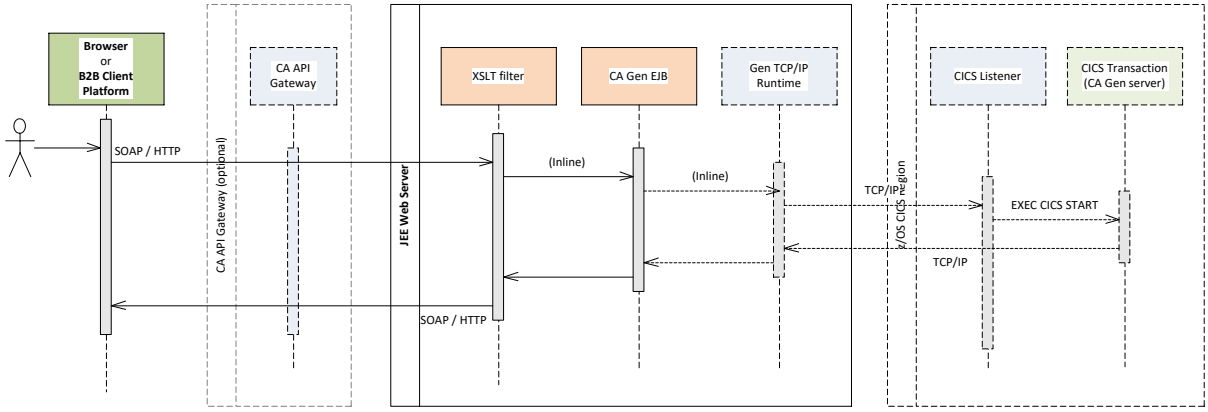

Figure 8 Typical EJB Web Service flow

Creation

To create an EJB SOAP Web Services suitable for deployment to a JEE application server, the following approach is taken:

1. In the Developer toolset, the target procedure step is selected, and server EJB code is generated, targeting EJB Web Services[7]
2. That code is then "built" (or compiled and packaged) using the CA Gen Build Tool
3. Using CA Gen Studio
    a. Interface of the Web Service is customized, using the Procedure Step Interface Designer
    b. Web Service is defined, to call the target server procedure step through its customized interface
    c. Web Service custom interface is then generated and built
4. With the CA Gen Build Tool, the server and its custom Web Service is then assembled
5. The web service and the server EJB are deployed to the JEE Application Server.

Processing

5. An HTTP(S) request, with a SOAP payload, is sent to the Web Server, by the WS consumer
6. Incoming message is transformed, using the XSL transformation corresponding to the customizations brought to the imports
7. The corresponding EJB is then called, which processes the input and sends a message back, through the same channel.

Options

Flexibility can be added at both ends of the interaction:

- The EJB can act as a *façade* procedure step[8] to a server transaction, with as objective to:
    - Provide views of higher level than the implementation-level information sometimes used in tight coupling,
    - Do some pre- and post-processing, to fulfil specific Web Services requirements
    - Orchestrate multiple calls to CA Gen servers
    - ...

[7] EJB Web Services generation adds annotations to the generated Java code, for automatic generation of a basic web service, as per the EJB3 norm
[8] CA Technologies recommends this approach, as it gives much more flexible and resilient Web Services

- Publishing the Web Service to the CA API Gateway, as mentioned before.

CA Gen Web Service middleware-based SOAP Web Services

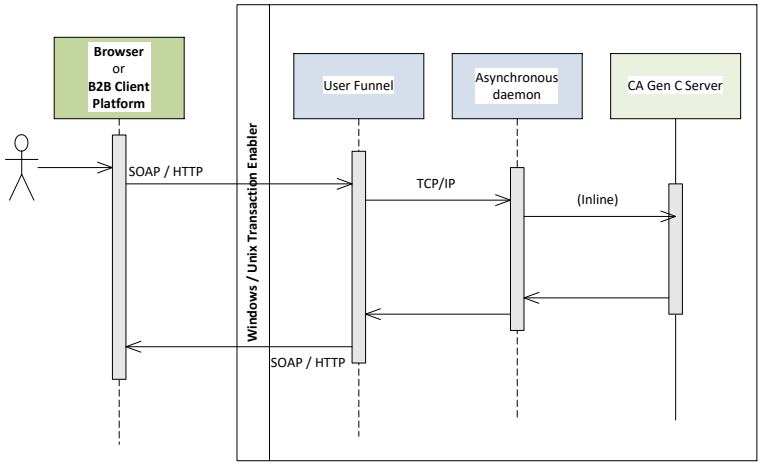

Figure 9 Typical Web Service middleware-based flow

Creation

There is no need for any special procedure: C servers generated for the Transaction Enabler automatically incorporate code for access as Web Services[9].

Processing

1. An HTTP(S) request, with a SOAP payload, is sent to the Transaction Enabler's User Funnel, by the WS consumer
2. Through the Asynchronous Daemon, it is transferred to the target server, which decodes the XML payload into import views, then calls the relevant procedure
3. Upon return, the export views are converted into a SOAP reply by the server, then sent back to the invoker.

Option

Flexibility can be added at both ends of the interaction, by publishing the Web Service to the CA API Gateway, as mentioned before.

[9] As of PTF RTN85015, RTX85303, RTI85303, RTR85303 and RTL85303, this feature can be disabled at runtime (mainly for security reasons)

CA API Gateway-based REST Web Services

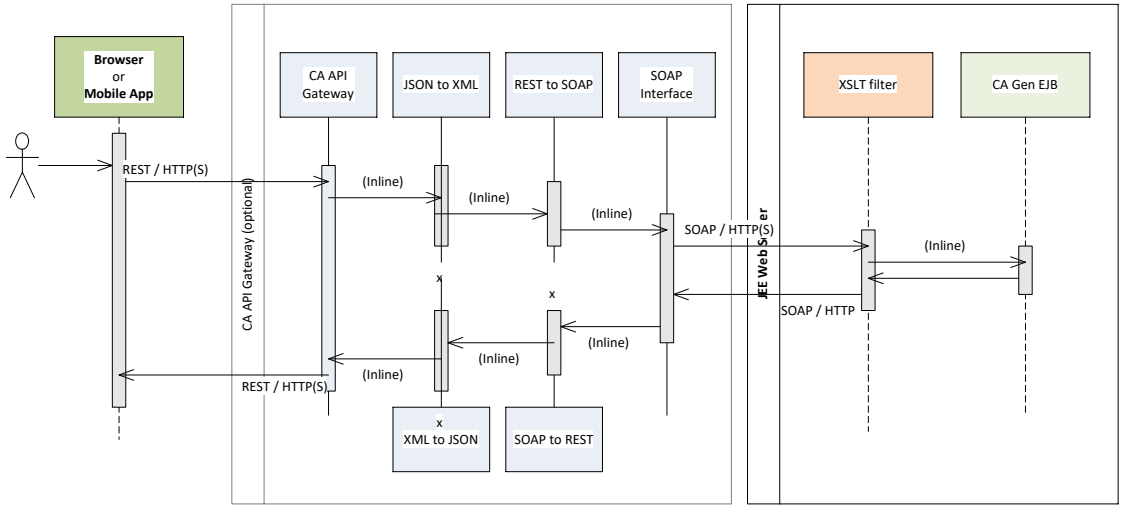

Figure 10 Typical REST flow through API Gateway

Creation

To access a CA Gen SOAP Web Service (here, an EJB-based Web Service as an example) using REST, through the CA API Gateway, the following approach is taken:

1. The CA Gen SOAP Web Service is created, as described in the previous sections
2. Using the CA API Gateway Policy Manager:
    a. The Web service is published
    b. If JSON is used as messaging format, conversion of the incoming message from JSON to XML is specified
    c. Conversion of additional information (query parameters, HTTP method, …) to appropriate elements / attributes of the XML message is also specified
    d. XML message conversion back to JSON is then specified…
    e. As well as error reporting

Processing

1. A REST HTTP(S) request, with a JSON / XML payload, is sent to the CA API Gateway, by the WS consumer (typically a mobile app)
2. The Gateway converts the JSON message and request information to XML,
3. It then invokes the CA Gen SOAP Web Service
4. Upon return, the information is transformed back from XML to JSON, then sent back to the invoker.

Option

Flexibility can be added at the Gateway level, as mentioned before.

Web API Designer for CA Gen-based REST Web Services

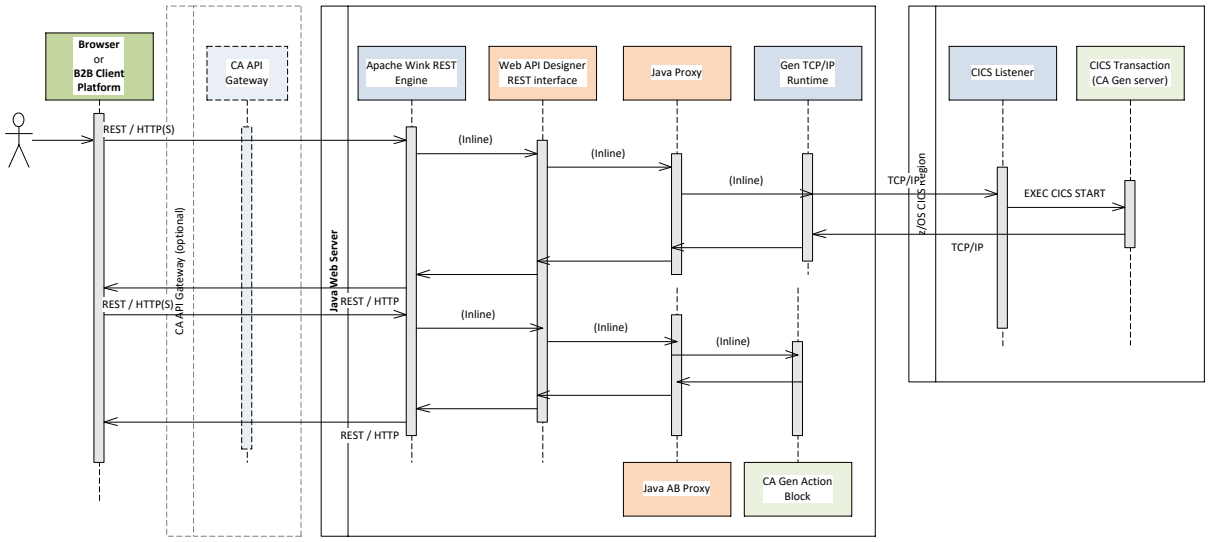

Figure 11 Typical REST flows through Web API Designer for CA Gen

Creation

To access services of a resource, implemented as CA Gen code, using REST Web Services, the following approach is taken:

1. The CA Gen procedures / Java action blocks implementing the services are generated for their respective target environment
2. Using the Web API Designer for CA Gen wizard:
    a. The REST resource is defined, mapped to a CA Gen entity, specification or interface type, its attributes mapped to CA Gen attributes, and services mapped to CA Gen procedure steps / action blocks
    b. Error handling, service verb, and mapping from incoming information to import views and from export views to outgoing information is further specified
    c. Code and documentation are then generated and built, then packaged into a REST application
3. The REST application Web Archive is then deployed to the target Web Server
4. If a remote server needs to be invoked, the CA Gen runtime configuration file is adapted to specify the location and middleware used to access the remote server.

Processing

1. A REST HTTP(S) request, with a JSON / XML payload, is sent to the Web Server, by the WS consumer (typically a mobile app)
2. The REST engine unmarshalls the payload and calls the interface corresponding to the request,
3. That interface converts the incoming information into import views, and invokes the relevant CA Gen piece of logic
4. Upon return:
    a. Export views are used to populate the relevant objects in the interface,
    b. The resulting information is marshalled to XML/JSON by the REST engine
    c. Then sent back to the consumer.

Option

Flexibility can be added at the Gateway level, as mentioned before.

CA Gen Web Service middleware-based consumption of SOAP Web Services

This feature is only mentioned for completeness. As, at this stage at least, it's not possible to invoke external Web Services through the middleware, it's currently not considered as an integration mechanism.

CA Gen External call to SOAP Web Services

The newer *CALL EXTERNAL* statement works this way:

1. Optionally creates worksets, attributes and views per the WSDL of the SOAP Web Service,
2. Converts action block import views into a SOAP request,
3. Invokes the SOAP Web Service
4. And converts the SOAP response to export views.

Canam SOA Composer-based call SOAP Web Services

Canam Software works quite similarly to the CA Gen CALL EXTERNAL, except that the call is made from an external action block.

The major difference with the CA Gen native call is, according to Canam's web site, that SOA Composer is not limited to HTTP, but can also access SOAP Web Services through WebSphere MQ.

Refer to Canam Software (http://www.canamsoftware.com/Home.aspx) for documentation about SOA Composer.

CBD / Limited Microservices Architecture with CA Gen

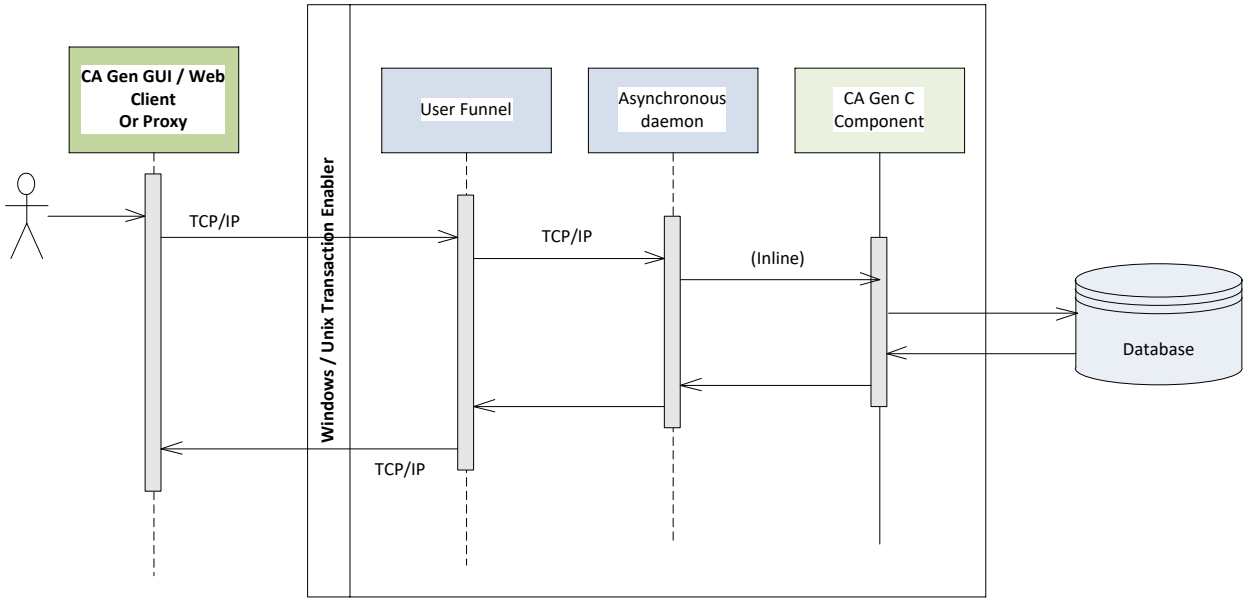

Figure 12 Limited Microservices Architecture

This architecture is named limited, as it uses proprietary-only interfaces and optional middleware. This is the kind of architecture many CA Gen customers have implemented and deployed, at least partially.

Creation

To create a limited Microservices architecture, the following steps are executed:

1. *In provider models*:

   a. Create a specification layer, made up of entity / specification or interface types and associated transactional operations definitions
   b. Develop the bridge to interface specification to implementation layer and back, using type maps or ad hoc mapping
   c. Package the operations into one or a few server load modules (transactional operations)
2. *In consumer models*:
   a. Migrate the component operations definitions (server load modules or libraries in short expansion) into the model
   b. *On the client side*: invoke transactional operations by invocation from client-side code (GUI, Web or Proxies)
   c. *On the CA Gen server / online side*: develop an external action block to consume transactional operations through proxies.

Processing

1. *On the client side*: when a transactional operation is invoked:
   a. The CA Gen client runtime fetches the location of the operation through the adequate configuration file (Client Manager Directory Services, commcfg.properties, commcfg.ini or commcfg.text)
   b. It then directs the request to the relevant server and waits for the reply
   c. Upon receipt of the reply, it resumes operation of the consumer;
2. *On the CA Gen server / online side*:
   a. The external action block code instantiates the proxy, fills in import information and invokes it
   b. The CA Gen runtime acts then as described in points b and c above
   c. The external action block captures the exported information into the relevant consumer views
   d. Consumer server / online logic resumes operations

Full Microservices Architecture with CA Gen

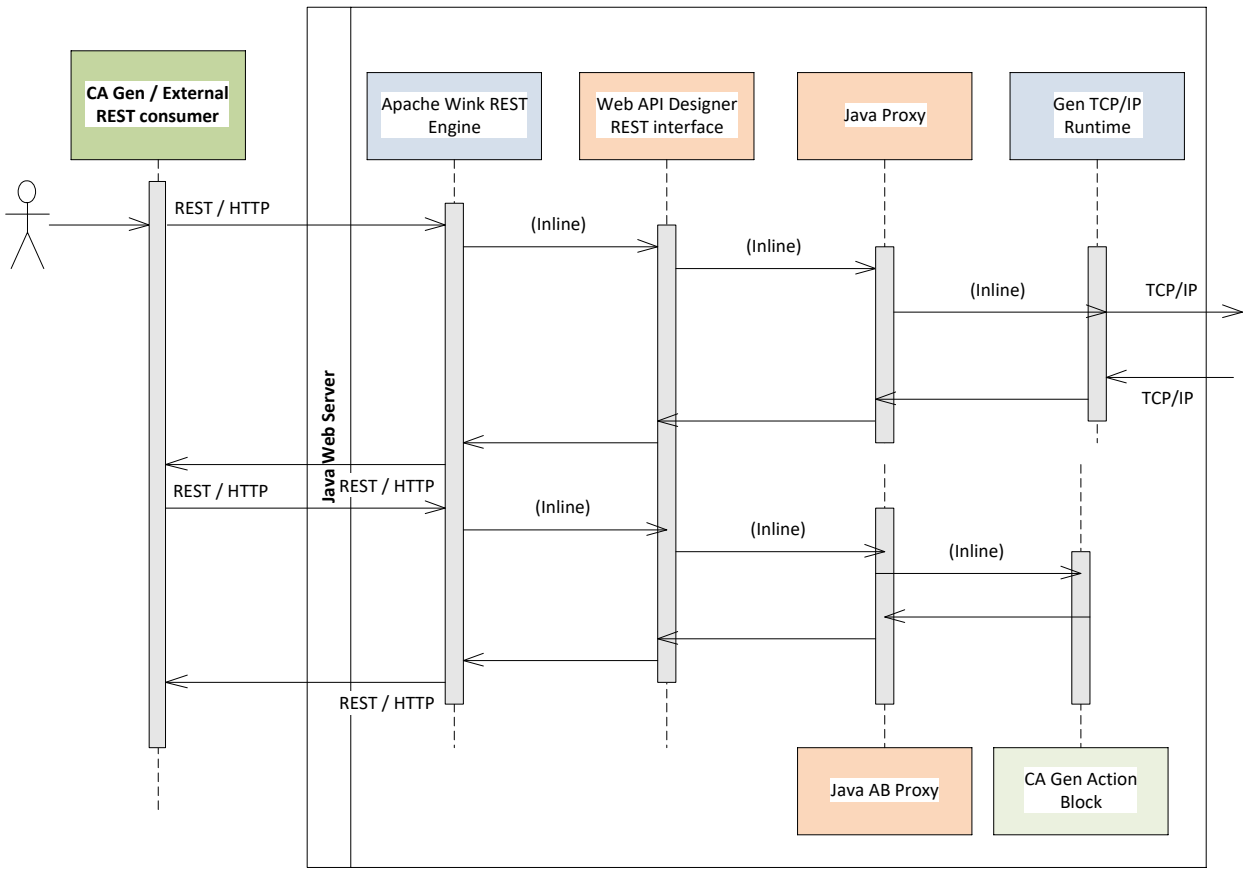

Figure 13 Full Microservices Architecture

This is the full Microservices Architecture, systematically consuming published transactional operations as microservices through the REST protocol.

Creation

To create a full Microservices architecture, the following steps are executed:

1. In *provider models*:
    a. Create a specification layer, made up of entity / specification or interface types and associated transactional operations definitions
    b. Develop the bridge to interface specification to implementation layer and back, using type maps or ad hoc mapping
    c. Package the operations into one or a few server load modules (transactional operations)
    d. Develop external action blocks to make use of some public library or 3rd-party product (for instance; gSOAP for C code, Apache CXF for Java, DB2 REST user-defined functions for Cobol, …) to invoke the microservice[10]
    e. Package the external action blocks into an Operations / z/OS library;
2. In *consumer models*:
    a. Migrate the component operations definitions (Operations / z/OS library containing the external action block definitions) into the model
    b. In the CA Gen code, invoke the microservice by USEing the external action block;
3. In *external consumer code*:
    a. Use any available REST library / 3rd-party solution to develop interface code from the Web API Designer-generated documentation.

Processing

1. The consumer code (CA Gen / external) invokes the local interface code to send an HTTP request to the REST engine running on a Web Server
2. The REST engine unmarshalls the message into the appropriate constituents and invokes the relevant Web API Designer REST interface code
3. The REST interface code instantiates the proxy, fills in import information and invokes it
4. As the Web Server is an integral part of the deployed Microservice, responsibility for specification of the target CA Gen server location is moved from the consumer (in the limited approach) to infrastructure of the Microservices
5. The Microservices platform then makes sure the message is properly forwarded to the business server, captures its reply, and sends it back, correctly marshalled, as an HTTP response to the consumer code.

Option

Additional security (authentication, availability…) can be added to the architecture by protecting the REST engine server with an instance of the CA API Gateway.

[10] When the intention is to interface with components from CA Gen code, it's normal to centralize the development of such external action blocks to the team who develops the component, as those external action blocks will become the *de facto* interface to the component at the CA Gen level

Summary

With CA Gen, your developments are future-proof, probably even more than what you think!

This white paper has presented the current integration possibilities offered by CA Gen and analyzed in more depth the solutions that, a priori, would best suit most needs of most customers.

It has also, very briefly, presented some ways CA Services can help achieve integration objectives.

The services offered by CA Services are certainly not limited to White Papers like this one, or to the automation of activities. They cover the whole spectrum of services, somehow related to CA Gen, from knowledge transfer workshops and high level architectural guidance to execution of very specific tasks requiring deep skills, from tool usage, through process guidance to improvements and automation, from facilitation of interactions between different functions, through assistance to the development process, to infrastructure management (administration, upgrades, ...).

Appendix – Comparison of Web Services solutions

Comparison of Web Services provisioning solutions						
Solution	**Axis Web Services**	**.Net Web Services**	**EJB Web Services**	**WS Middleware**	**API Gateway-based REST**	**Web API Designer**
Language	Java	C#	Java	C	Java	Java (+ Cobol/CICS, based on demand)
Required licenses	Java proxies	.Net Proxies	EJB generation	Transaction Enabler	CA API Gateway + relevant Gen license	Web API Designer + one of - Java proxies, - EJB generation or - Web generation and Java AB Proxy
Flexible	No	No	Minimal	No	Yes	Quite
Protocol	SOAP/HTTP(S)	SOAP/HTTP(S)	SOAP/HTTP(S)	SOAP/HTTP	REST/HTTP(S)	REST/HTTP((S)

[i] Software AG, Annual Report 2008 http://www.softwareag.com/corporate/images/SAG_GB_eng_BUCH_300309_final_schutz_tcm16-51535.pdf