Guide to
Creating CICS Web Services
CICS applications continue to provide the foundation for many critical business processes in the enterprise. The original developers of these applications did not know that someday a need to integrate them with web, cloud or mobile applications would arise. But the functionality these CICS applications provide – as well as their performance and reliability – makes it appealing and necessary to make their functionality available to middle-tier applications.

**Advantages of the Web Services Approach to Integration**

Developing CICS web services has become the favored approach for enterprises to integrate the data and business logic of CICS applications with any web, cloud or mobile applications. Here’s why:

- **For those coming from a mainframe or CICS background**, a web service provides a standard way to integrate distributed applications using XML, SOAP, WSDL and UDDI open standards over an IP backbone. Just about any application that doesn’t run on the mainframe can easily exploit a web service.
- **For those who are distributed systems developers**, CICS applications have historically emitted data on 3270 screens, or used IBM protocols and interfaces to share data. This makes them challenging integrate targets. Creating CICS web services keeps distributed developers and systems from having to gain mainframe expertise to create integrations.

CICS web services are developed with the HostBridge JavaScript Engine (HB.js), a server-side JavaScript engine that runs within CICS. The development is done using the HostBridge Eclipse Integrated Development Environment (IDE), an Eclipse plug-in that provides developers a way to easily and rapidly develop and deploy scripts/web services from mainframe resources. This approach has many advantages over trying to rewrite CICS applications, or using screen scraping technology for integration:

- **The business logic of the CICS applications is unaffected and fully exploitable by middle-tier applications.**
- **No changes to CICS applications are required.**
- Using JavaScript, CICS web services are easy to develop and fast to deploy.
- **The web services approach to CICS integration is standards-based.**
• Integrations built via the web services approach are high-performing and scale well.
• CICS web services operate within existing mainframe security infrastructure.

Enterprises that use JavaScript to create a CICS API or web service are simplifying the integration of CICS applications with anything in the middle-tier. The core functionality of these CICS applications is easy to exploit through integrations that improve productivity, and to enable the agile creation of new, innovative web, cloud or mobile applications. **In the course of doing so, no changes to the host applications are necessary, nor do the integrations rely on screen-scraping technology.**

For example, a **major Asian bank** adopted the HostBridge JavaScript Engine (HB.js) for CICS web services creation. After the initial training, productivity soared. The bank’s mainframe development team was able, in just an hour, to create a CICS web service to enable a new international funds transfer application. The original development estimate for completing this project was six months!

**How to Create CICS Web Services**

Developing CICS web services using JavaScript is a fast, easy process with a shallow learning curve. It takes little time to gain proficiency, and with just a little experience, web services creation can occur very quickly.

**Step 1: Understand the Business Process**

The first step in creating a CICS web service is to understand the underlying business process for which you want to turn into a web service.

What is the purpose and objective of the business process? You should
gain a thorough, complete understanding of the business process before writing the first line of code for your web service. Using JavaScript to create the web service is generally easy when the business process is well understood.

For example, a common business process for an automobile insurer is adding a vehicle to a policy. Understanding this business process requires knowing the required inputs and the required outputs. What types of responses does the application expect? Similar business process examples include checking an account balance or changing an address. Understanding the business logic and flow of inputs and outputs is the first step to creating a web service to integrate the application.

**Step 2: Identify the Error Conditions**

After you have a good understanding of the business process, and know the expected inputs and outputs, identify the error conditions.

What can go wrong with inputs or responses to the CICS application that supports the business process? When creating a CICS web service, the service will act as a human operator would to interact with the application. If the integration target is a CICS COMMAREA* application, then the interaction is simply a straightforward API call. At other times, however, the interaction is with a BMS or other CICS application that was designed to interact with a human through a 3270 device. It is therefore important to understand the error conditions that an experienced operator knows intuitively.

Understanding the error conditions that can occur enables you to develop a web service that anticipates and handles these conditions.

**Step 3: Select the Interaction Style**

After you identify the error conditions, decide what style of interaction you want the CICS web service to support.

The great thing about using web services to support integration is all of the standards-based options are typically available for styling the integration.

*In CICS, the Communications Area (COMMAREA) remains a common way to pass data between traditional CICS application programs. The HostBridge JavaScript Engine, because it runs within CICS, can access data in the COMMAREA.
interaction between the middle-tier and the CICS application(s) via the web service. XML, SWAGGER, JSON, SOAP, RESTful and many other standard protocols are possible choices. Choose the approach that best meets your environment and requirements. It is relatively easy to support multiple styles via a web service, allowing you to alternate input methods for the same web service, if desired. It is a best practice to make sure that web services are written so that you can put a different input or output interface on it at a later date.

Determine what style of interface you want the web service to present to the middle-tier before you begin writing the JavaScript code.

**Step 4: Write Your Test Cases**

After you’ve determined how to style the interaction between the middle-tier and CICS applications, write your test or unit cases.

Write test cases to test all the inputs and outputs and the error conditions that you identified in the previous step. Your test cases should cover all aspects of what you need the web service to deliver. By creating test cases, you also make sure that changes you make won’t cause downstream errors or problems. Furthermore, as you add new features and functions to the web service in the future, or choose to use it other environments, you have a way of testing it to make sure it performs as expected. It’s good programming practice to have a way of testing your application as you write it and after you make any changes to it. There are several tools available to aid in the writing of test cases, such as SoapUI and Postman, which has become the favorite means of the HostBridge technical services team for testing web services.

**Step 5: Write the Web Service**

After the test case is written and well defined, and your web service passes all the tests, then it is time to write the web service.

Host or mainframe developers, even though they may not have a background in JavaScript development, have a shallow learning curve. They are therefore excellent candidates for developing CICS web services because they usually understand the host applications. With JavaScript, developers
have the option to create procedurally-oriented or object-oriented web services. Either option is fine.

Writing the web service is not difficult or technically demanding, because the developer is simply following the application flow. The human operators, who in many cases have been sitting at consoles for years executing these flows, are excellent sources for understanding the application flow. Writing these flows, adding the error conditions and checks is usually pretty straightforward. Developers will write some JavaScript code for the web service, test it, write some more code, and test it again. The HostBridge Eclipse IDE makes this development process simple.

When developing CICS web services, it is a best practice to create discrete, small, identifiable processes instead of trying to create a comprehensive web service that is all encompassing. When you write these fine-grained services, they are more reusable, security is less of an issue, and they’re much easier to write. Most developers discover that they are better off writing five fine-grained services, for example, than to write one, big web service that does a lot of different things.

**Step 6: Deploy the Web Service**

After writing and testing a web service you’ve written, the final step in the process is to deploy it.

Deploying CICS web services developed with HostBridge is very simple: it’s a mouse click to deploy it onto the host. Once on the host, your Quality Assurance team and process can complete testing, and once finished, move it into production. HostBridge complies with and supports any standard change control practices you may have in place. As a web service you’ve created moves from one environment to the next, the process doesn’t circumvent any of the controls that are in place. The process this guide describes for developing CICS web services integrates with any change control system or process you have in place.

**How Long Does it Take?**

Developing web services as a means of integration or modernizing CICS applications is neither complex or time-consuming.
The process is simplest when the target application is a CICS COMMAREA program. The web service just needs to pass the expected parameters directly and get the output. Very little checking is required, because the CICS program was probably designed to do most of the checking itself. When the inputs for and outputs from the COMMAREA program are understood, it can take as little as two or three hours to write a web service to integrate it.

The strength of HB.js and the web services approach is for integrating terminal-oriented CICS applications. For example, if the business process and supporting CICS application requires navigating 15 or 20 screens, and the navigation path diverges based on the input provided or the output received, the developer can orchestrate all the screen navigation through the web service. Even with complex screen navigation, when the inputs, outputs, and error conditions are well known, a developer who is proficient with JavaScript and uses the HostBridge Eclipse IDE can write a complex web service in about a week.

The experiences shared by HostBridge customers show that the learning curve for web services develop isn't very steep, and proficiency is gained quickly.

See How it Works

To see how the web services creation process described in this document works, visit the HostBridge home page and click the “See a Demo” button. You’ll be shown a short video that presents the CICS web service creation process.

Try it in Your Environment

The best way to learn how to create CICS web services, and to understand how they perform in your environment, is to conduct a HostBridge Pilot Application Workshop (PAW). For U.S.-based customers, there is no cost to doing a PAW, and for international customers, only travel costs for HostBridge staff are incurred.

To learn more about how your organization can create and exploit CICS web services, contact the HostBridge team or visit the HostBridge website.