WebSphere® software

Using containers and channels to enhance CICS interprogram data transfer.

By Steve Wood, Chartered Engineer
IBM CICS product development, IBM Software Group

September 2005

Contents

Introduction

For many organizations that run IBM CICS® systems, their CICS applications conduct their core business – and represent a significant intellectual asset, developed over many years. However, new styles of programming have evolved that enable Web-based business models. If organizations want to embrace these new business models without sacrificing reliable, core business transactions, CICS applications must be capable of cooperating with these new styles of programming. New programming styles often require the exchange of large amounts of data, rather than efficiently managed exchange of parameter data between CICS programs.

Now, IBM CICS Transaction Server, Version 3.1 introduces a new approach, called *containers* and *channels*, that is designed to provide a straightforward and flexible mechanism to exchange large volumes of structured parameter data between CICS programs. By using this approach, you can enable CICS programs to easily exchange unlimited data with virtually any Web-based program. The containers and channels approach removes the constraint of the 32KB communications area (COMMAREA) limit. It also promotes easy linkage between the valuable and time-proven core business processes of your enterprise and the new business models that can extend the core business to help you achieve competitive advantage.

This paper examines the benefits of new interfaces introduced by CICS Transaction Server for z/OS, Version 3.1. These interfaces are optimized to exchange rich and varied structured data content among CICS application components. This paper explains when these interfaces can be applied to gain best-possible advantage.

Pushing the limits on data exchange

For most organizations that run CICS systems, CICS applications are fundamental to the core value proposition of the business. This means that reuse and integration of your CICS applications with new solutions makes sound business sense. However, new solutions are often required to use very different data-exchange methods from the efficient, strongly typed methods used by traditional CICS applications.

Modern enterprise solutions typically use a loosely coupled style by defining clear component boundaries and minimizing dependency on shared resources among the system elements. This style offers the advantage of increased flexibility to accommodate future business requirements. Many newer solutions are designed to exchange the entire information payload in the form of XML documents. Parameter data described in XML has very different characteristics from the optimized data formats familiar to CICS programmers.

By design, XML is extensible, so you can add further data elements when application requirements change. XML structures can accommodate a varied mix of data types, and it is common for an XML document to reference large binary objects such as images, along with character and numeric payload data.

To help make the structure easier for programmers to understand and work with, the XML tags that delimit the document structure elements typically have meaningful names. As a result, XML is verbose; the tag description of the structure format and bounds of the elements within it can often occupy a significant proportion of the content of the document. When you add in these lengthy tag descriptions, XML-based parameter data areas can require large areas of memory when passed by value between program elements. Frequently, the 32KB limit imposed on the traditional CICS COMMAREA is not enough to accommodate modern applications.

Passing data with traditional methods

As Figure 1 shows, CICS applications have traditionally passed request parameters and sent response parameters through the `EXEC CICS LINK` application programming interface (API) that specifies a single area of memory called the *COMMAREA*. The COMMAREA contains the typed parameter data values in the structure defined by the applications, and is limited by the API to no more than 32KB in length.

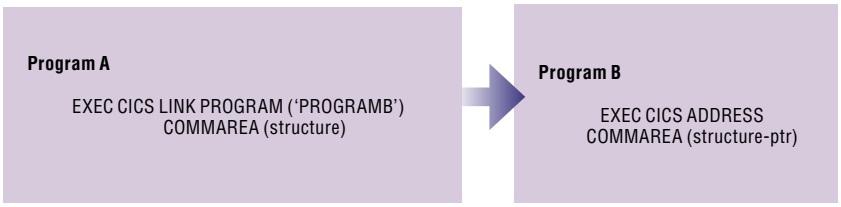

Figure 1. Passing data between applications using the COMMAREA

Applications using the COMMAREA can be written in COBOL, PL/I, C, C++, Java™ and other languages. In a similar way to subroutines, CICS COMMAREA programs are unaware of which application invoked them or how they were invoked. They are often *stateless* – that is, they do not have any record of previous interactions. CICS Transaction Server automatically manages the transactional and security contexts that are typically inherited from the caller and transaction definition.

For historical reasons, CICS applications have been highly optimized to be memory efficient, and have been easily able to pass the parameter data required within the size constraints imposed by the CICS API on the COMMAREA. When application components had to share more than 32KB of data, programmers used managed shared-systems resources, such as Virtual Storage Access Method (VSAM) files. The containers and channels approach provided in CICS Transaction Server for z/OS, Version 3.1 provides an alternative style of implementation for the exchange of large quantities of parameter data.

Some COMMAREA issues with handling large data objects

- Applications must use a circumvention technique, such as using external VSAM files, or splitting the data into separate parts. This method increases risk, as well as programming time and effort.
- Passing XML documents by value throughout the request process path becomes inhibited because the size constraint applies to calls between CICS programs both within the local system and between CICS systems, to parameter data passed between CICS tasks, and to external client programming interfaces such as external CICS interface (EXCI) and the CICS client external call interface (ECI).
- Data structures used to define a COMMAREA payload can become overloaded. Redefining structures on the same area of memory (dependent on flags or whether the input, output or error information is passed) increases the risk of program errors. Similarly, confusion about the validity of fields can result in application-programming errors.
- An overloaded COMMAREA structure increases transmission time between CICS regions because the structure size must account for the maximum size of the data that could be returned by the called program—and this parameter size depends on the request logic invoked. CICS Transaction Server must always allocate memory to accommodate the return of the maximum COMMAREA structure size, even if the parameter content has not changed during the call processing, which compromises memory management.
- A code-page conversion of COMMAREA structure is complex because binary and character data can't be easily separated.

CICS programs are typically grouped into application suites, or components, consisting of a number of programs designed to perform some business action on data that is common to these programs. In most cases, some programs within a component provide external access to clients or end users. These programs use a public interface for the component that is distinct from the private interfaces provided by each of the programs within a component. Because both public and private interfaces are implemented as COMMAREAs, many of the application assets that you have created over the years can be reused as the building blocks for components. This capability is key to the effective reuse of applications.

Today, implementing modern business solutions means that more and more CICS applications are integrated with contemporary enterprise solution components that exist outside the CICS environment. As a result, CICS applications must increasingly process large quantities of structured parameter data, in both XML and non-XML formats. The 32KB constraint imposed on COMMAREA size compromises the ability to extend CICS applications to new enterprise solutions, and limits the reuse of core CICS applications.

The CICS solution

CICS Transaction Server, Version 3.1 introduces a new approach that provides an easy and more-flexible mechanism to exchange large volumes of structured parameter data between CICS programs. This new approach is provided by containers and channels.

Containers

A *container* is a uniquely named block of data that can be passed to a subsequent program or transaction. It refers to a particular parameter data structure that exists within a collection of virtually any form of application parameter data. You can (and should) choose a container name that is a meaningful representation of the data structure housed. For example, in a medical application, the container name might be `<prescription>`. CICS Transaction Server provides EXEC API verbs to create, delete, reference, access and manipulate a container as well as to associate it with a channel.

A container can be any length, and container size is constrained only by the available user storage in the CICS address space. It can include data in any format required by an application. An application can create any number of containers as required, and can use separate containers for different data types, such as binary and character data. This capability helps ensure that each container structure is based on a unique area of memory. It also minimizes the potential for errors that commonly arise when parameter data for multiple applications is overloaded on a single memory area by isolating different data structures and making the association between data structure and purpose clear. You can read more about discipline in program implementation in the section of this paper entitled, "Good practice for programming containers and channels."

Channels

A channel is a uniquely named reference to a collection of application parameter data held in containers. It's analogous to a COMMAREA but it's free of its constraints. You can choose a channel name that is a meaningful representation of the data structures that the channel is associated with. For example, again, in a medical application, a channel name might be `<patient history>`. This collection of application parameter data serves as a standard mechanism to exchange data between CICS programs. CICS Transaction Server provides an EXEC API that associates a named channel with a collection of one or more containers — offering an easy way to group parameter data structures to pass to a called application. CICS Transaction Server removes a channel when it can no longer be referenced (when it becomes out of scope).

The benefits of the containers and channels approach

To simplify programming, the CICS system – rather than the application – controls the life cycle and scope of containers and channels. CICS Transaction Server helps ensure that data integrity is preserved and storage resources are managed to meet the processing requirements of the workload.

Application programmers – and the applications that they write – benefit from:

- *An unconstrained, CICS technology-supported method of passing parameter data.*
- *Segregation of parameter data structures, each part represented by a named container structure.*
- *A loosely coupled approach, with very little dependency between components involved in the interchange, so that each can be used freely with other components.*
- *The freedom to dynamically determine the nature of the passed data and to select the appropriate processing required.*
- *A CICS standard API for optimized data exchange between CICS programs implemented in any CICS technology-supported language.*
- *Ease of parameter passing by using unique named references.*
- *Ease of understanding by using named references for parameter payloads.*

Along with these application programming benefits, the CICS implementation of containers and channels is optimized for performance, by efficient memory management and data transfer, and for data conversion between different code pages, on a per-container basis so that only the required containers are moved or translated.

Performance

The internal CICS implementation that supports the container and channel APIs is optimized to enable efficient memory management and data transfer between the calling applications, for both local and remote calling. The CICS implementation recognizes and tracks both modifications that are made to container content and the creation of new containers. CICS Transaction Server helps ensure that only the necessary new and modified containers are transferred between the calling applications. As a result, if one parameter data structure resides in a single container, the calling applications can benefit from complete access to the data content in all containers that are in scope. Furthermore, the performance of the calling mechanism is optimal because the path length necessary to move data is minimized.

Explicit code-page conversion

CICS Transaction Server for z/OS, Version 3.1 includes functionality that enables data conversion of container content. This function can mean converting between character and binary formats or between different character-set encodings. You invoke the conversion by specifying a coded character set identifier (CCSID) as a parameter when accessing a container. For example, a CCSID can cause CICS Transaction Server to convert the data content into Extended Binary Coded Decimal Interchange Code (EBCDIC). This capability is useful when CICS Transaction Server must receive data content from applications implemented in an environment that is not natively EBCDIC, such as distributed systems or Java 2 Platform, Enterprise Edition (J2EE) application servers. Associating conversion operations at the container level helps simplify how applications handle code-page conversion for different data structures because you can apply different code-page conversions to separate containers even if these containers are associated with the same channel.

Good practice for programming containers and channels

The ability to segregate parameter data into different types is fundamental to making containers and channels a best-practice approach for replacing the traditional COMMAREA structure. Depending on the nature of the application data structure, you might have to create separate containers for:

- *Each parameter structure in the copybook*
- *Read-only and read-write parameter structures*
- *Input and output parameter structures*
- *Binary and character data structures*

Separating data types for reliability

An application can create as many containers as required, and it can use separate containers for different data types, such as binary or character data. This practice helps ensure that each container structure is based on a unique area of memory. By helping to ensure isolation between different data structures and clear association between data structure and purpose, you are less likely to experience the errors that can commonly arise when parameter data for multiple applications is overloaded on a single memory area. These characteristics of isolation and association with purpose promote more rigorous discipline in program development, with improved reliability.

You can associate a set of purpose-related containers with a channel. This approach provides an easy means to group parameter data structures. Separating parameter data structures into discrete containers allows functional loose coupling between the components of an application suite. You can see the advantage of this approach when you introduce new capabilities to an application suite, because it enables progressive implementation of the new capability. If you upgrade a service provider with new logic, the parameter data related to new function can be separated in a new container. The additional logic can then test for the existence of the new container, and if it's present, perform the new action. Because existing functionality and data is unaffected, and the change is self-contained, the potential for error is reduced.

You can immediately deploy the change, and the existing behavior stays the same, because the service provider could not be invoked using the new container and new parameter data. This approach promotes flexible solution implementations because you can progressively introduce service requestors that pass the new parameter data to exploit the new function. As a result, you can avoid a single wholesale functional change throughout the system – reducing implementation risk and introducing the opportunity to stage the introduction of a new capability.

Programming actions with containers and channels

CICS Transaction Server provides EXEC APIs for containers and channels in all supported CICS programming languages. CICS also provides Java CICS (JCICS) classes that enable you to use containers and channels to exchange data between CICS J2EE applications and traditional CICS procedural applications. API verbs enable you to `PUT` and `GET` containers (see Figure 2) and, in the process, to associate a container with a channel and to delete containers. Another `EXEC CICS` API verb makes it possible for you to `MOVE` containers from one channel association to another. A channel can be passed on LINK, `START`, `XCTL` and `RETURN` commands, and is supported on a distributed program link (DPL) system. You can use the `START CHANNEL` command to initiate transactions remotely. The `INQUIRE` API verb enables applications to discover and process the appropriate containers and channels, including `ASSIGN CHANNEL` and `STARTBROWSE`, `GETNEXT` and `ENDBROWSE CONTAINER`.

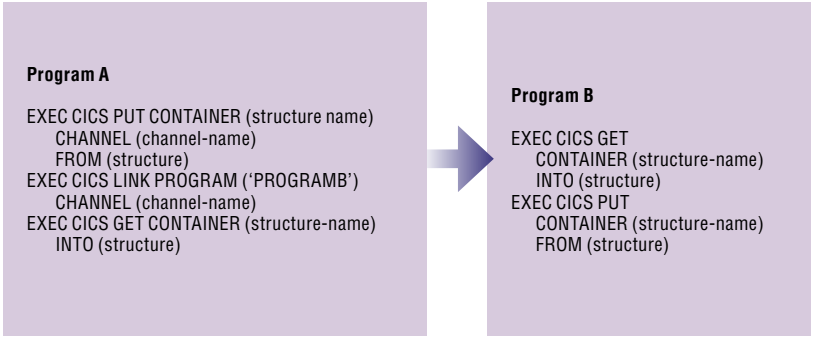

Figure 2. Passing data between applications using containers and channels

Benefits of the containers and channels approach

- Offers a more flexible and intuitive alternative to the COMMAREA. By using separate containers for logically different data, the containers and channels approach helps simplify language structures and minimize the impact of changes to the interface.
- Enables large amounts of data to be passed between CICS applications because they aren't subject to the 32KB restriction. You are only limited by the amount of storage you have available.
- Requires minimal application changes.

The programming verbs provided by CICS Transaction Server to support containers enable you to specify whether to put binary (bit) or character (char) data into a container. This capability helps CICS Transaction Server to optimize data-processing conversion because it can recognize binary content and avoid any code-page conversion. If an application needs to convert any character data between different code-page representations, you can specify a CCSID as a command option. Simply specify an optional CCSID parameter on the `PUT` container and `GET` container verbs supported by CICS Transaction Server and the required code-page conversion is applied as the data is moved to or from the container. You can apply different code-page conversions to different structures based on criteria such as data type or intended usage.

To find the full set of EXEC CICS API verbs and options for containers and channels in the "CICS Application Programming Reference" topic of the CICS Transaction Server for z/OS Information Center, visit http://publib.boulder.ibm.com/infocenter/cicsts31/index.jsp.

Data-separation example

CICS application-error information is only optionally returned when an error occurs, by definition. Separating this information into a discrete container can simplify application logic. The `GET CONTAINER` command for the error container returns a `NOTFOUND` condition. You can use this condition as a clear logic switch for the main or error-processing path. Using this condition also reduces the potential for the application to misinterpret the returned data. By creating this condition, you can also improve the clarity of the documentation of the error handling within the program implementation. And the application benefits from improved transmission efficiency between CICS regions because the error container only needs to be sent when an error occurs.

A strategic approach to containers and channels

The containers and channels approach complements other CICS technologies for On Demand Business solutions, such as Web services support, and presents a new opportunity to implement new, well-constructed business logic that can exchange and process large volumes of rich, structured data in CICS applications. You can best use this new capability where:

- *New function is required for XML processing in Web services solutions.*
- *New structured data styles expose constraints in existing CICS applications.*
- *Existing CICS application suites must be optimized to process large data volumes.*

The simplest approach to converting existing CICS applications to take advantage of containers and channels might seem to be to replace an existing COMMAREA implementation by creating a new channel with a single container that holds the existing COMMAREA structure. However, this approach does not enable you to separate a monolithic parameter block into its separate component parts. Consequently, the application suite can't fully take advantage of the benefits of the containers and channels approach—including the ability to separate data structures and optimize data passing, described previously in this white paper.

In many cases, it is effective to maintain existing CICS programs using the original COMMAREA techniques. IBM recommends that, if you're evaluating the containers and channels approach, you carefully select new or existing applications to help ensure manageable risk and high returned value as you adopt new capabilities.

With the right interfaces to business-logic programs in CICS Transaction Server, your existing CICS assets can be reused in many On Demand Business solutions. They exploit and complement IBM z/OS® operating system qualities of service, such as high availability and scalability at a low cost per transaction, with excellent security. The new containers and channels capability in CICS Transaction Server for z/OS, Version 3.1 demonstrates how CICS Transaction Server continues to lead the way in delivering maximum return on investment (ROI) while minimizing risk.

Summary

Traditionally, CICS programs have used COMMAREAs to exchange data. To overcome limitations in the size of data passed, CICS Transaction Server for z/OS, Version 3.1 provides an improved method of exchanging data involving two new concepts: containers and channels. *Containers* are essentially named parameter data structures that, when grouped together into sets called *channels*, are analogous to a COMMAREA.

The containers and channels approach has several advantages over COMMAREAs:

- *Containers can be any size and, as a result, can extend beyond the maximum 32KB size of a COMMAREA. There is no limit to the number of containers that can be added to a channel, and the size of individual containers is limited only by the amount of storage available.*
- *A channel consists of multiple containers, enabling it to be used to pass data in a more structured way. In contrast, a COMMAREA is monolithic block of data.*
- *Unlike COMMAREAs, channels don't require the programs that use them to know the exact size of data returned, making programming easier.*

Channels can be used by CICS application programs written in any of the CICS technology-supported languages. For example, a Java client program on one CICS region can use a channel to exchange data with a COBOL server program on a different region.

The ability to use multiple containers reduces the complexity of designing programs, because now programs don't have to reformat data into a single COMMAREA. And multiple containers also allow greater independence when maintaining programs. In the past, every program that called a utility with a large COMMAREA had to be recompiled when data elements were added to the COMMAREA. Now, when multiple containers are used, only programs affected by the addition of data elements need to be recompiled.

Through the loose coupling with other applications, the new containers and channels approach is the last element needed to enable CICS programs to take part freely in the composable processes that are such an important part of On Demand Business. With Web services capabilities, the new unlimited data exchange capabilities in CICS Transaction Server mean that CICS applications can be quickly integrated with any other application, fully aligning your core business transactions with the new business models that are defining today's enterprise.

For more information

To learn more about the containers and channels approach and IBM CICS Transaction Server for z/OS, Version 3.1, contact your IBM representative or IBM Business Partner, or visit:

ibm.com/cics

IBM United Kingdom Limited
Hursley Park
Winchester
Hampshire
SO21 2JN
United Kingdom

Produced in the United States of America
09-05

G224-7535-00