IBM WebSphere Application Server for z/OS Version 8.5

Optimized Local Adapters

See the WP101490 White Paper for more details

What is "WOLA?"

The acronym "WOLA" stands for WebSphere Optimized Local Adapters. WOLA is a function of WebSphere Application Server for z/OS that allows very fast, efficient, and low-latency memory to memory exchanges between Java applications in WAS z/OS and programs running in external address spaces such as CICS, IMS and Batch[1]. WOLA is an excellent means of communication between programs where high throughput is required.

Why WOLA?

The following bullets summarize the value of WOLA:

- ***Fast*** -- the exchange is performed by copying messages from one memory location to another. Few things are as fast as that. That translates to *low latency*, which helps in high-volume environments where each unit of latency tends add up.
- ***Efficient*** -- because WOLA avoids network or other exchange protocols, it tends to use relatively less CPU per exchange. Again, in high-volume environments any savings per exchange tends to add up.
- ***Secure*** -- because no network is used for the exchange, there is no exposure to network security concerns. The exchange is completely secure, known only to sender and receiver.
- ***Flexible*** -- WOLA may be used for outbound calls from WAS into CICS or IMS, *or* for inbound calls from those two (or batch programs) into WAS. WOLA is bi-directional. That provides greater architectural flexibility for your solutions.

Technical Highlights

- First available with WAS z/OS Version 7.0.0.4, it continues to be a key feature of WAS z/OS V8.0 and V8.5. It is part of WAS z/OS; no additional license charge applies.
- Functionally enhanced several times since its introduction. See the "History of Updates to WOLA" section of the WP101490 Techdoc at `ibm.com/support/techdocs`.
- Bi-directional exchanges:

Outbound	Java in WAS z/OS invokes the non-Java program running in the external address space. A common usage pattern is Java in WAS invoking a program in CICS.
Inbound	The program outside of WAS invokes the Java program running in WAS z/OS. A common usage pattern is a COBOL batch program invoking a Java program in WAS.

- Transactional integrity using z/OS RRS

CICS	Two-phase commit outbound WAS into CICS (7.0.0.12)[2] [3] Two-phase commit inbound CICS into WAS (7.0.0.4)
IMS	2PC outbound WAS into IMS over OTMA (8.0.0.5 | 8.5.0.2)[4] 2PC inbound IMS into WAS (8.0.0.4 | 8.5.0.1)

- Security Assertion

CICS	Outbound: ID on WAS execution thread Inbound: Region ID or application user ID
IMS	Outbound: ID on WAS execution thread Inbound: ID on IMS thread

- Programming Model

Outbound	• Java program in WAS write to the Common Client Interface (CCI) of the supplied WOLA JCA resource adapter • Target program in CICS may be entirely shielded from any WOLA awareness with supplied WOLA Link Server Task that installs into CICS region. • If desired, programs in CICS may bypass Link Server Task and implement WOLA APIs. • Target program in IMS may be entirely shielded from any WOLA awareness when communication over OTMA
Inbound	• Target Java in WAS must be stateless EJB and use supplied WOLA class libraries when implementing interfaces • Programs in CICS, IMS or Batch would use the WOLA APIs to invoke target program WAS

- WOLA APIs -- native APIs are provided to provide programming interface for cases where APIs called for. Supported languages are: COBOL, C/C++, High Level Assembler, and PL/I.
- "Bridge" Programs -- these are Java or non-Java programs that implement WOLA requirements and shield other programs from having WOLA awareness. This is a common development pattern.

  For example, when invoking *inbound* the stateless EJB that implements using the WOLA class libraries may then turn and invoke *other* EJBs locally. Those EJBs don't need to know about WOLA. The WOLA-aware EJB serves as a "bridge" to other EJBs.
  Similarly, a program in CICS may use the WOLA APIs to invoke Java programs in WAS, with *other* CICS programs using normal CICS DPL to drive the WOLA bridge program.
- CICS Channels and Containers support available in limited function format in 7.0.0.4, and enhanced to full function in 8.0.0.5 and 8.5.0.2.

  The original support was limited to a single fixed-name channel with a single container. The full function provided in 8.0.0.5 | 8.5.0.2 provides multiple channels, multiple containers and mapped records.
- Round-robin distribution across multiple WOLA connections into separate CICS regions 8.0.0.1 | 8.5.0.0.
- IMS large multi-segment messages support starting in 8.0.0.0 or 8.5.0.0.
- SMF 120.10 records for outbound calls WAS into CICS or IMS.

Enabling WOLA in WAS z/OS

The effort to enable WOLA in the WAS z/OS environment is relatively simple. See the WP101490 Techdoc at `ibm.com/support/techdocs` and locate the "Quick Start Guide." That guide will take you step-by-step through the process.

1 The supported external address spaces are: CICS, IMS, Batch, USS, and ALCS
2 The version and fixpack numbers indicate when the capability first came available
3 This required CICS TS 4.1 or higher
4 WAS z/OS V8 and V8.5 are concurrently available at this time

Implementing WOLA in CICS

To use WOLA in a CICS region requires a few relatively simple system programmer tasks to update the region CSD and concatenate the WOLA libraries to the CICS `DFHRPL DD` statement. These steps are clearly spelled out in the WP101490 "Quick Start Guide" found at `ibm.com/support/techdocs`.

WOLA CICS Link Server Task

The supplied WOLA Link Server Task provides a means of shielding target CICS programs from awareness the WOLA interface, as well as providing an infrastructure to handle transaction and security assertion from WAS into CICS.

The Link Server Task by default carries of name of `BBO$`. It may be started manually using a supplied `BBOC` transaction, or started at CICS region initialization using either a PLT program or CICS sequential terminal support.

You have considerable flexibility to customize the names of the WOLA components within your CICS region.

WOLA JCA Resource Adapter

WOLA comes with a JCA resource adapter named `ola.rar`. It installs into the WAS z/OS runtime just like any other JCA resource adapter. Once installed, it enables the creation of connection factories to which your Java applications connect to communicate over WOLA *outbound* to CICS, IMS or even batch jobs.

In this sense WOLA becomes like any other JCA resource. The programming model is the same. Some of the methods require WOLA-specific information, but otherwise it is a standard JCA interface.

WOLA and IMS

WOLA works with IMS as well:

- Outbound over OTMA: this requires no change to the target IMS applications, and no enablement of WOLA in IMS. Java in WAS z/OS uses the WOLA JCA resource adapter, which maps the request to OTMA.
- Outbound using WOLA APIs: applications may write to the WOLA APIs and receive calls from Java in WAS over ESAF[5]. IMS customers refer to this support as "WOLA Direct[6]."
- Inbound using WOLA APIs: applications may write to the WOLA APIs and invoke Java assets over ESAF into WAS z/OS.

WOLA works with Message Processing Programs (MPP), Batch Message Processing (BMP), IMS Fast Path (IFP), and Batch DL/I applications.

WOLA and Batch

An interesting challenge has long been how to leverage Java assets running in WAS z/OS as part of a traditional z/OS batch job process. With WOLA a high-speed, low-latency connection option is available.

From your batch job (COBOL, High Level Assembler, C/C++, PL/I) you use the supplied WOLA APIs to register into the WAS z/OS server and invoke the target EJB. The programming model can be as simple as three APIs: `BBOA1REG` to register, `BBOA1INV` to invoke the target EJB, and `BBOA1URG` to unregister when the batch job completes. More sophisticated use of the APIs is possible as well if your program requirements call for it.

5 External Subsystem Access Facility

6 We can't take credit for this phrase, but we like it! ☺

See the WP101490 Techdoc at `ibm.com/support/techdocs` and locate the "Native API Primer." That guide will take you on a step-by-step tour of the APIs and make clear how they're used.

Real-Life Use Case Examples

- Business function formerly run in CICS was re-engineered to run in WAS z/OS. WOLA was used to communicate to remaining CICS function. Offload from the General Processor usage to z/OS Specialty Engines (zAAP or zAAP-on-zIIP) by moving workload from CICS to WAS z/OS more than offset CPU usage of WAS z/OS infrastructure.
- COBOL billing program needed to access vendor tax calculation package running on distributed platform. WAS z/OS served as the multi-threaded web services engine. COBOL batch program used WOLA and asynchronous control to keep 150 WOLA connections into WAS z/OS busy with requests for tax calculation over web services to remote server. 8,000 transactions per second achieved.
- WAS z/OS serves as call consolidation point into a CICSplex environment where core business resides. WOLA is used to communicate to two CICS gateway regions using the multiple WOLA connection and round-robin support provided in 8.0.0.1 and 8.5.0.0. High-volume aggregation of remote calls achieved with dual-path (and fault tolerant) high-speed connections into CICSplex.
- Business logic written in Java required complex high-speed transaction coordination with business logic in CICS. WAS on remote platform using web services transaction support tried but ruled out due to latency and overhead. WOLA with global transaction coordination between WAS z/OS and CICS with z/OS RRS as the transaction synchpoint coordinator proved effective and efficient for the business needs.

Summary

WOLA provides another architectural design tool at your disposal. When your design requirements call for very high speed, high-volume and low-latency calls between a WebSphere Application Server environment into a core z/OS business function such as CICS, IMS or Batch ... consider WAS z/OS and WOLA.

***WOLA**: Fast, efficient, secure and flexible.*

More Details

See the WP101490 Techdoc at `ibm.com/support/techdocs`. That Techdoc page has a number of documents related to WebSphere Optimized Local Adapters:

- *Quick Start Guide* - step-by-step guide for enablement and usage of WOLA in WAS z/OS and CICS
- *History of Updates* - overview of functional enhancements provided to WOLA over time
- *Overview and Usage Presentation* - with illustrative charts and speaker notes explaining key concepts
- *Native API Primer* - step-by-step tour through the native APIs with sample programs illustrating usage
- *Videos* - a series of narrated videos illustrating what WOLA is and how it is used

For more, contact Jeff Summers, Product Manager - WebSphere Application Foundation, at `summerje@us.ibm.com`

End of Brochure