

# The CICS Asynchronous API

Leigh Compton

IBM Washington Systems Center  
[lcompton@us.ibm.com](mailto:lcompton@us.ibm.com)

## Please note

IBM's statements regarding its plans, directions, and intent are subject to change or withdrawal without notice at IBM's sole discretion.

Information regarding potential future products is intended to outline our general product direction and it should not be relied on in making a purchasing decision.

The information mentioned regarding potential future products is not a commitment, promise, or legal obligation to deliver any material, code or functionality. Information about potential future products may not be incorporated into any contract.

The development, release, and timing of any future features or functionality described for our products remains at our sole discretion.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon many factors, including considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve results similar to those stated here.

# Abstract

Learn how the CICS Asynchronous API can help you reduce your application's response time. Available in the CICS TS 5.4 the Asynchronous API commands can help your applications become more robust, and allow you to make better use of your time.

Allow CICS to simplify your asynchronous processing mechanisms and break away from polling code to get better results with a first-class API.

Reduce the risk of unpredictable external services by calling multiple services and consuming the responses as soon as they arrive..

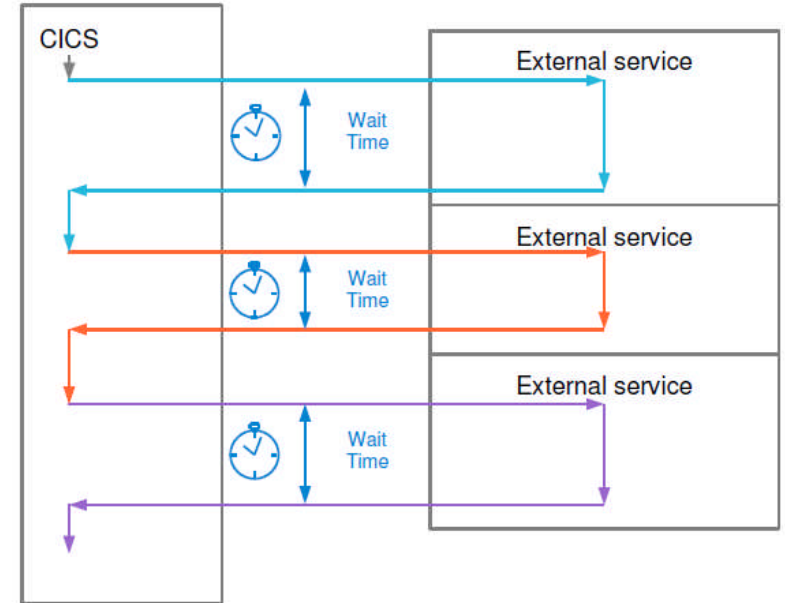
# Act 1

What customers are trying to do

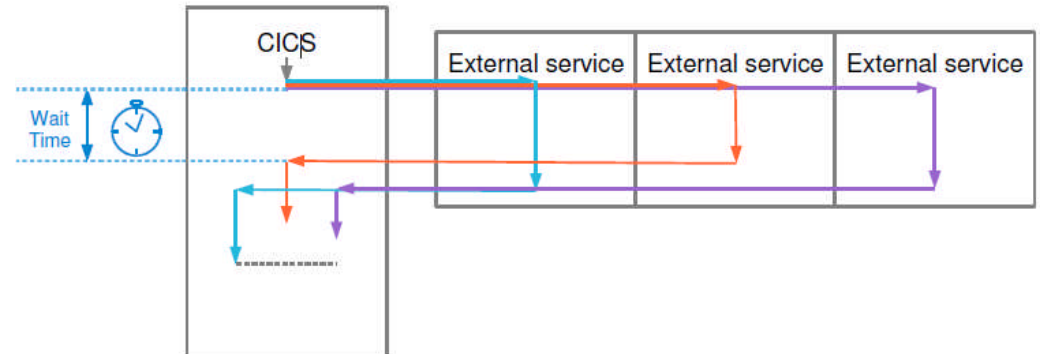
## Make better use of your time

Can you do something more useful than waiting around for a service call to return?

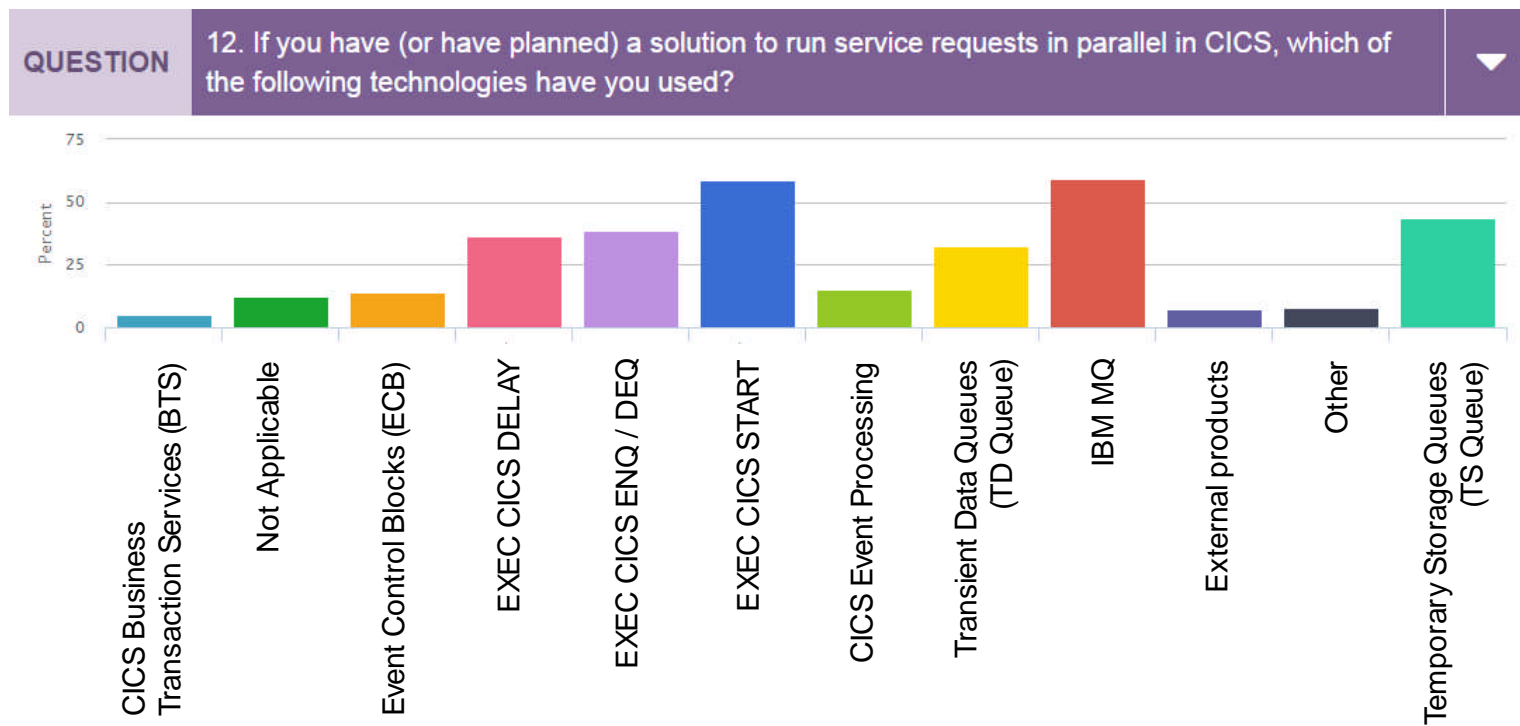
**From this:**



**To this:**



## How are customers doing this today with CICS?



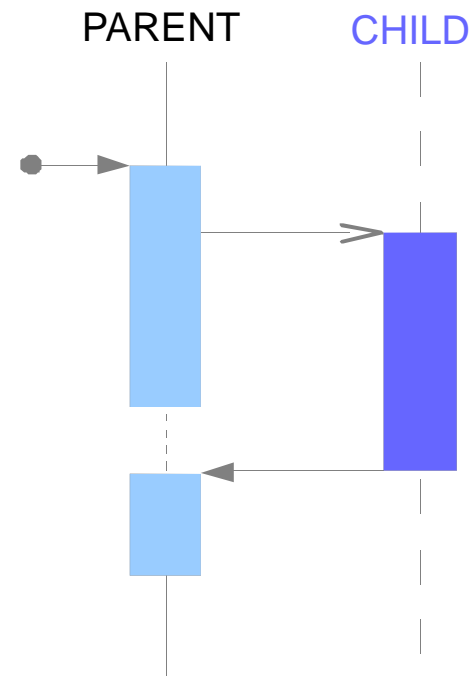
# Act 2

## A Better Solution

# New supported set of CICS Asynchronous API's

Key challenges

1. Run transaction asynchronously
2. Fetch child completion
3. Pass data safely

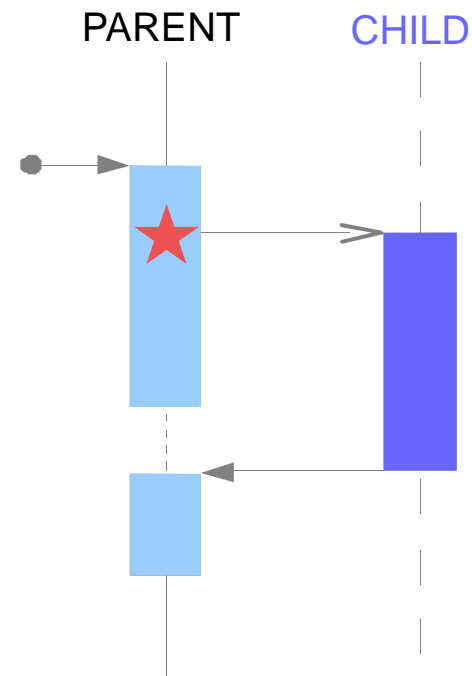




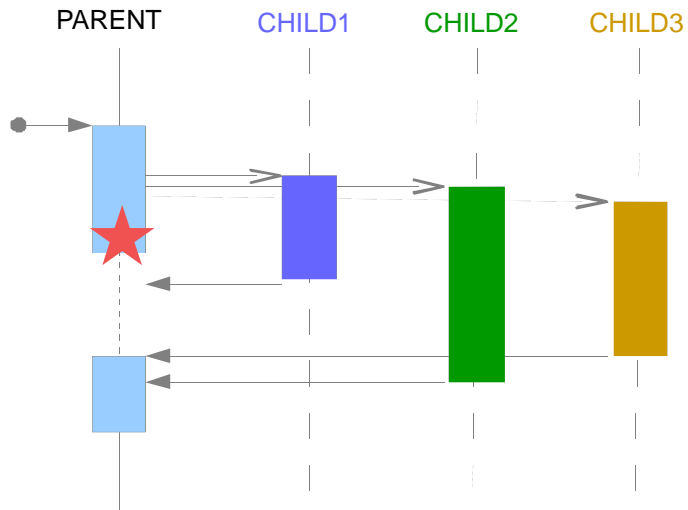
## Run transactions asynchronously

```
RUN TRANSID (transaction)
    CHILD    (identifier)
```

“I need a credit check but I can get on with some other work while I wait for the results”



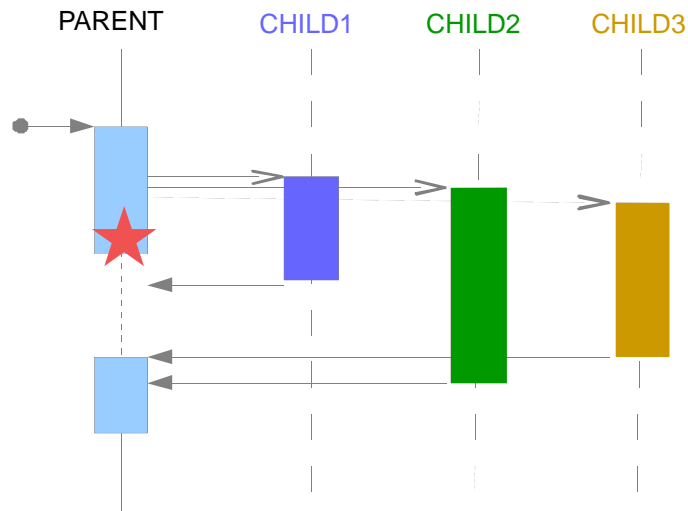
## Fetch child completion



```
FETCH CHILD(in-identifier)
```

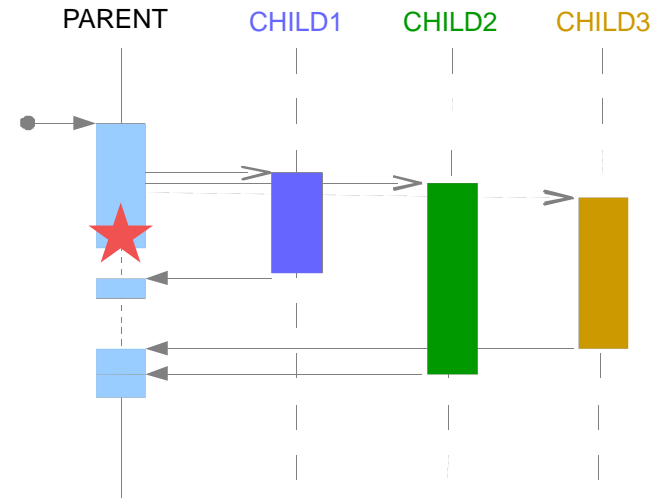
“I cannot continue processing the parent coordinator, until I have received confirmation from CHIL3”

## Fetch child completion – Two new commands



`FETCH CHILD(in-identifier)`

“I cannot continue processing the parent coordinator, until I have received confirmation from CHILD3”



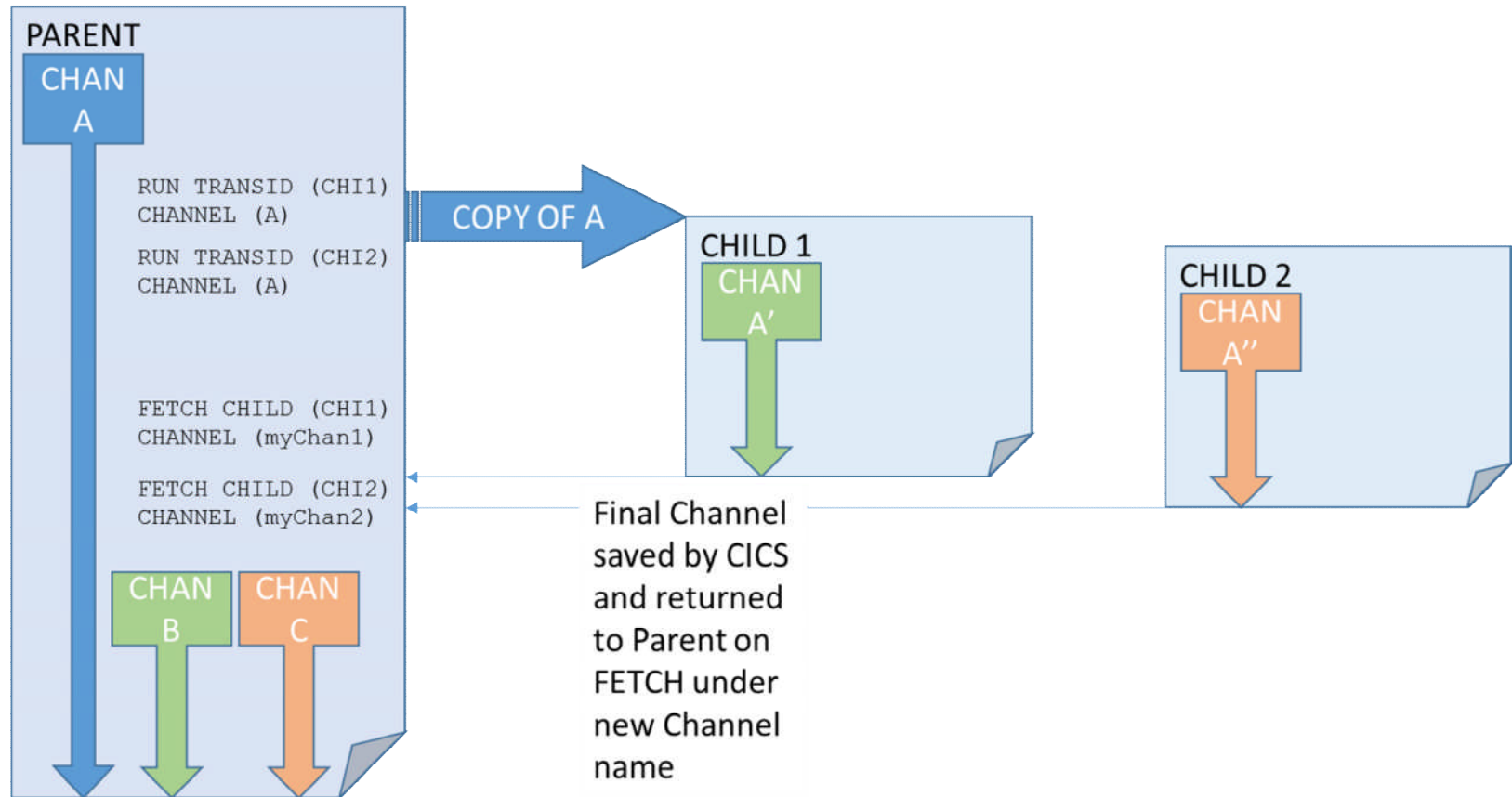
`FETCH ANY(out-identifier)`

“I only need the first one to reply”

# Passing Data

## CICS Channels and Containers

(not COMMAREA!)



# Technical Stuff:

## The programming interface

# RUN TRANSID

## RUN TRANSID

```
>>-RUN--TRANSID(name)--+-+-----+--CHILD(data-area)-----><
                               '-CHANNEL(name)-'
```

# RUN TRANSID

RUN TRANSID initiates a local child transaction that runs asynchronously with the parent transaction.

RUN TRANSID starts a task in the local CICS region, optionally passing a channel.

The child task inherits the security context of the parent task.

CHANNEL(name)

- specifies the name of a channel that is to be made available to the child task
- child is given a copy of the channel's containers

CHILD(data-area)

- specifies a 16-character binary data area into which CICS will place the child token.

TRANSID(name)

- specifies the local transaction to be started

# FETCH CHILD

## FETCH CHILD

```
>>-FETCH--CHILD(data-value)--+-----+----->
                                '-CHANNEL(data-area)-'

>--COMPSTATUS(cvda)--+-----+----->
                        '-ABCODE(data-area)-'

>--+-----+-----><
+-NOSUSPEND-----+
'-TIMEOUT(data-value)-'
```



# FETCH CHILD

FETCH CHILD command is used by a parent task to inquire on the status of a specific child task.

FETCH CHILD returns the status of the specified child task.

CHILD(data-value)

- specifies the child token to fetch the response from

COMPSTATUS(cvda)

- Returns a cvda indication completion status of the child task

CHANNEL(data-area)

- returns the 16-character name of the child task's reply channel.
- channel name is generated by CICS

TIMEOUT(data-value)

- specifies a fullword binary data value which is the maximum time in milliseconds that the command will wait for the child to complete.

NOSUSPEND

- Indicates that the command will return immediately without waiting for the child task to complete.

# FETCH ANY

## FETCH ANY

```
>>-FETCH--ANY(data-area)--+-----+----->
                                '-CHANNEL(data-area)-'

>--COMPSTATUS(cvda)--+-----+----->
                                '-ABCODE(data-area)-'

>--+-----+-----><
+-NOSUSPEND-----+
'-TIMEOUT(data-value)-'
```

# FETCH ANY

FETCH CHILD command is used by a parent task to inquire on the status of any child task.

FETCH CHILD returns the status of any completed child task which has not yet been fetched.

ANY(data-value)

- specifies a 16-character binary data area into which CICS will place the fetched child token.

COMPSTATUS(cvda)

- Returns a cvda indication completion status of the child task

CHANNEL(data-area)

- returns the 16-character name of the child task's reply channel.
- channel name is generated by CICS

TIMEOUT(data-value)

- specifies a fullword binary data value which is the maximum time in milliseconds that the command will wait for any remaining child to complete.

NOSUSPEND

- Indicates that the command will return immediately without waiting for the child task to complete.

# FREE CHILD

**FREE CHILD**

```
>>-FREE--CHILD(data-area)-----><
```

## FREE CHILD

FREE CHILD command frees a specified child token

Using the FREE CHILD command will free the resources associated with the child task when it completes, rather than waiting for them to be fetched.

CICS implicitly frees all child tokens when the parent task terminates.

CHILD(data-value)

- specifies the child token to be freed.

# JCICS classes

Not just a simple port!

Implements `java.util.concurrent.Future` interface

Three classes encompass all of EXEC CICS commands:

- `AsyncService`
- `AsyncServiceImpl`
- `ChildResponse`

`AsyncService`

`runTransactionID()`

`freeChild()`

`ChildResponse`

`getAny()`

`Future<ChildResponse>`

`get()`

`isDone()`

## JCICS example

```
final String childTransaction = "ABCD";
AsyncService myAsync = new AsyncServiceImpl();

Future<ChildResponse> myChild = myAsync.runTransactionID(childTransaction);

// Logic here to be processed while the child runs asynchronously

ChildResponse myResponse = myChild.get();
if (myResponse.getCompletionStatus().equals(CompletionStatus.NORMAL)) {
    System.out.println("Child task completed normally");
}
```

## Act 3

### The complete solution



What if my services are unreliable?

“How can I tell if the child completed successfully?”

The completion status is a first class parameter on the FETCH API

## How can I tell if my child completed successfully?

```
FETCH CHILD
>>-FETCH--CHILD(data-area)--+-+-----+----->
                                '-CHANNEL(data-area)-'
>--COMPSTATUS(cvda)--+-+-----+----->
                                '-ABCODE(data-area)-'
>--+-+-----+-----><
    +-NOSUSPEND-----+
    '-TIMEOUT(data-value)-'
```

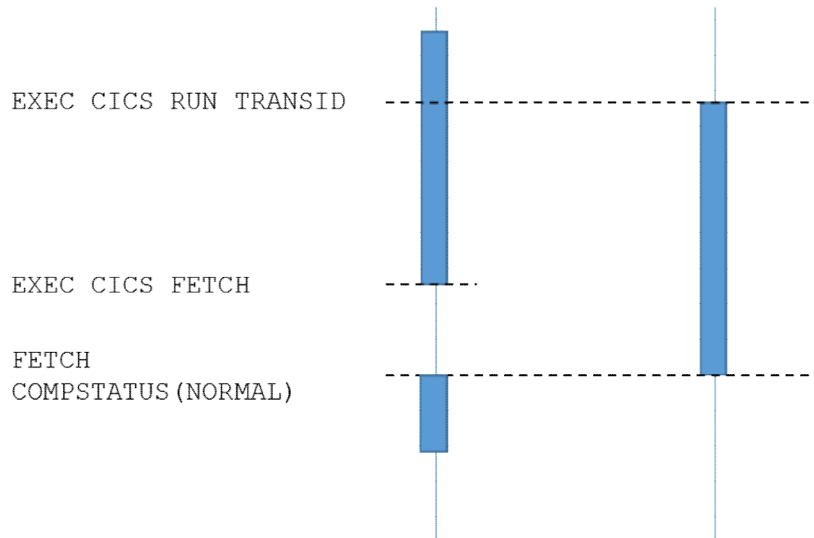
- COMPSTATUS returns a CVDA indicating the child task's completion status:
  - ABEND
  - NORMAL
  - SECERROR
- RESP=NOTFINISHED

## Don't want to wait forever!



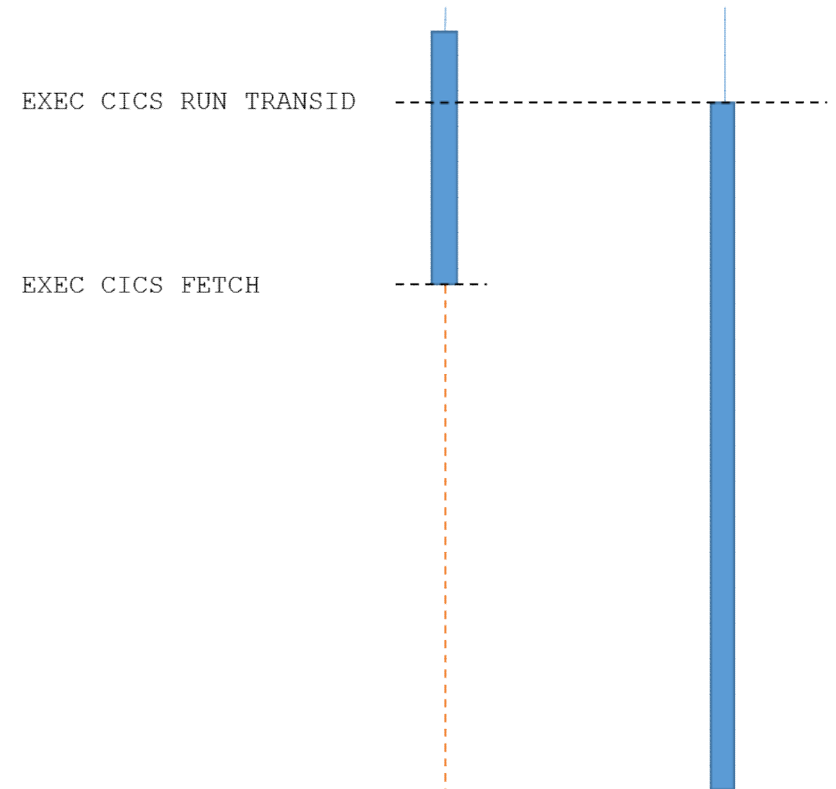
Parent

Child



Parent

Child

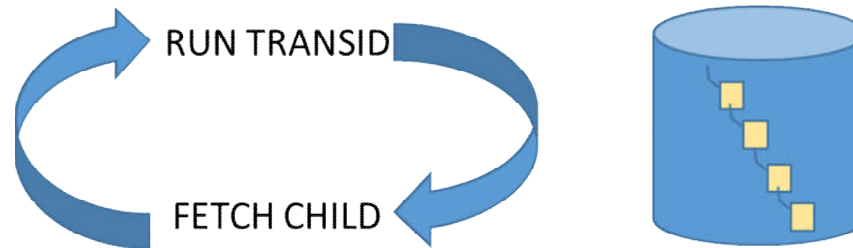


## Don't want to wait forever!

```
FETCH ANY
>>-FETCH--ANY(data-area)--+-+-----+-----+----->
                                '-CHANNEL(data-area)-'
>--COMPSTATUS(cvda)--+-+-----+-----+----->
                                '-ABCODE(data-area)-'
>--+-+-----+-----+-----+-----+-----><
    +-NOSUSPEND-----+
    '-TIMEOUT(data-value)-'
```

\* TIMEOUT(0) means that timeout is not being set

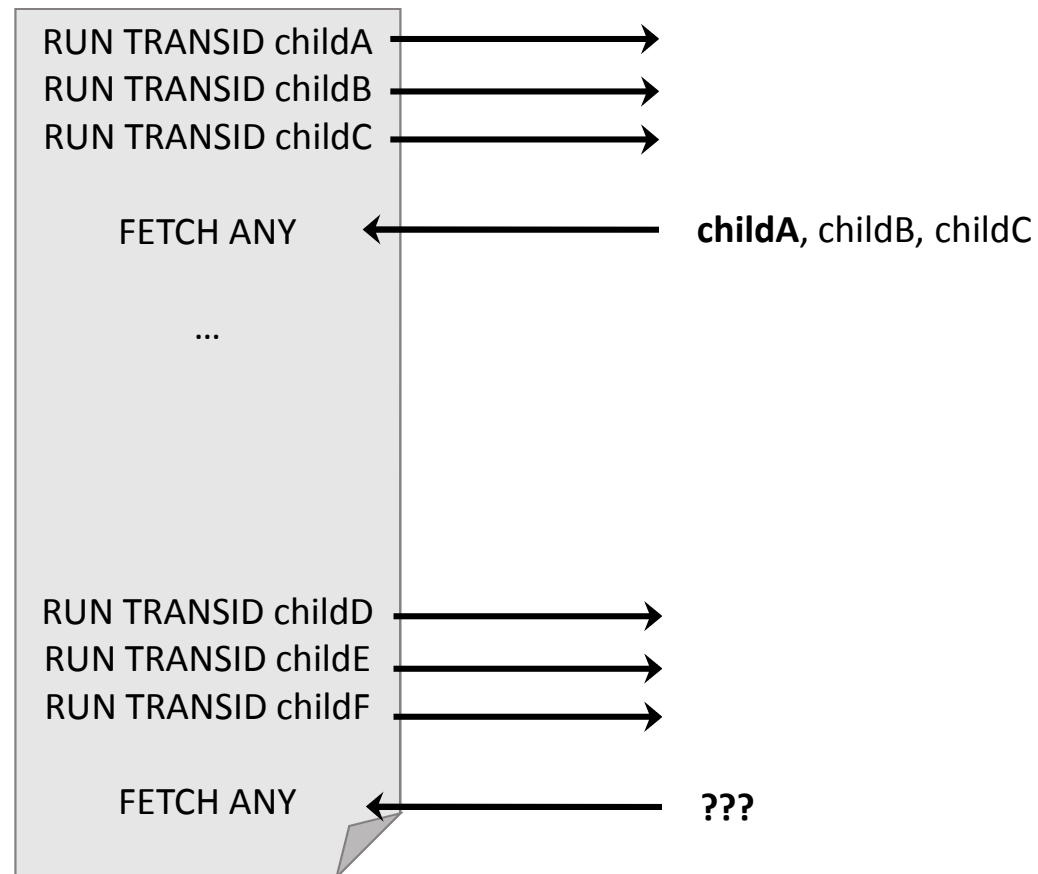
## FREE CHILD: Useful for Long-Running Parents



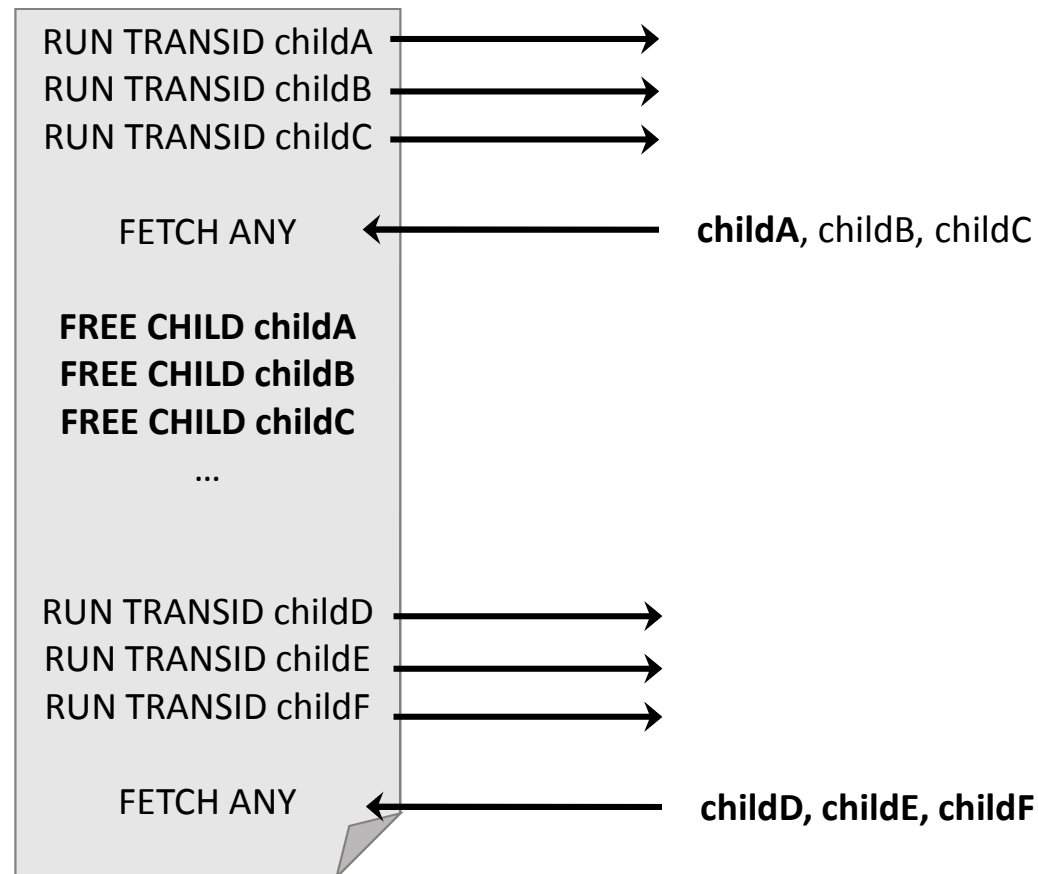
Ability to disassociate parent from child

Application can DELETE CHANNEL for already FETCHed results

## FREE CHILD: Batch Service Calls



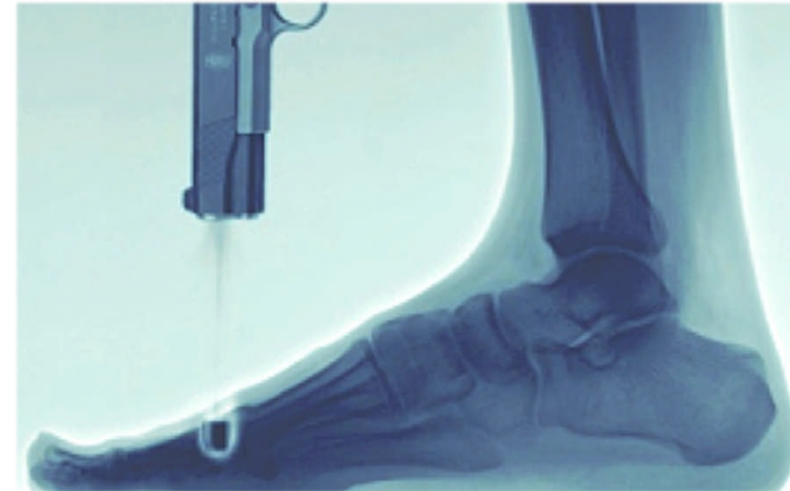
## FREE CHILD: Batch Service Calls



## CICS Automated Control

### Emergency Brake

- Control dispatching of asynchronous tasks
- Protect CICS rather than to optimise throughput
- Prevent flooding the CICS system
- Solely in CICS's control





## Transaction Classes

Control your systems!

**DO** have transaction classes

**DO NOT** put parents and children in the same transaction class

## CICS Policies

### Control your systems!

Task rule condition:

- RUN TRANSID
  - Threshold

Policy actions:

- issue message
- emit event
- abend task

# Articles, Knowledge Center, Example Code

github.com/**cicsdev**

## cics-async-api-fetch-child-example

A basic example demonstrating the passing of information from a parent to a child program using the CICS asynchronous API.

● C Updated on 5 Oct 2016

## cics-async-api-credit-card-application-example

An example application that compares calling services sequentially versus asynchronously, using the new CICS asynchronous API.

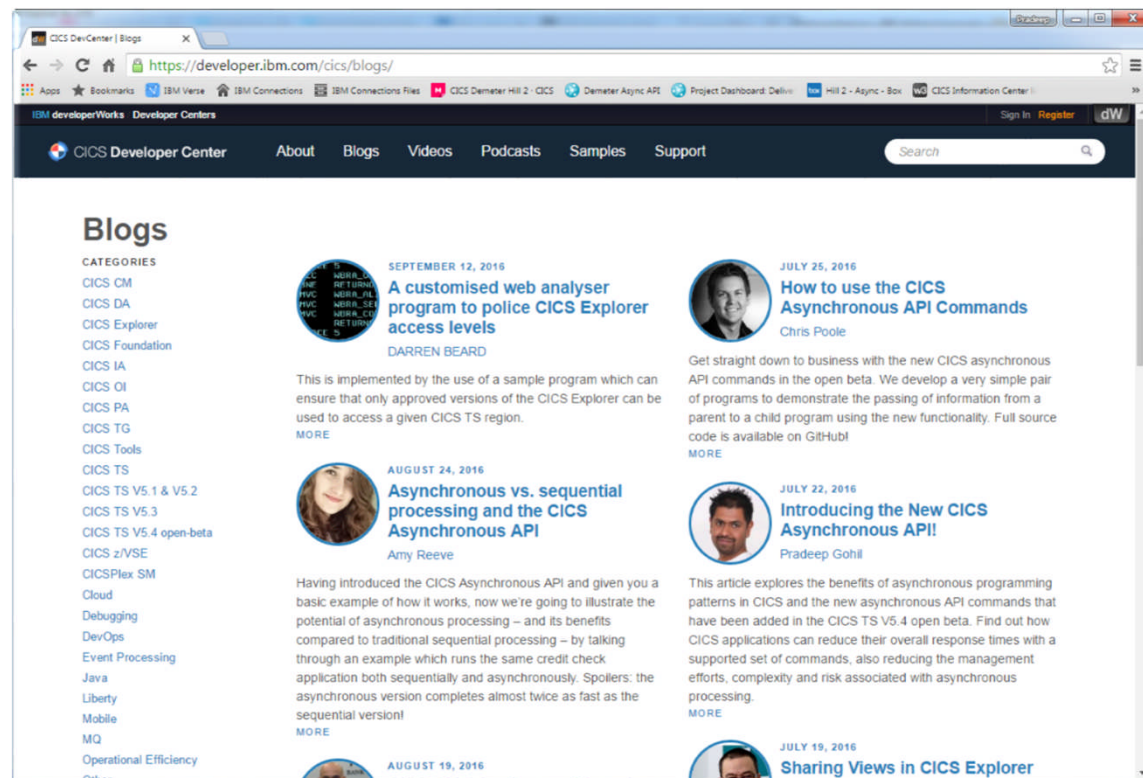
● COBOL Updated on 5 Oct 2016

## cics-async-api-channel-usage-example

Async API channel usage example

● Assembly 1 Updated on 20 Oct 2016

developer.ibm.com/**cics**



# IBM CICS Asynchronous API: Concurrent Processing Made Simple

## Redbook SG24-8411

<http://www.redbooks.ibm.com/abstracts/sg248411.html?Open>

This IBM® Redbooks® publication covers the background and implementation of the IBM CICS® asynchronous API, which is a simple, accessible API that is designed to enable CICS application developers to create efficient asynchronous programs in all CICS-supported languages. Using the API, application developers can eliminate the overhead that is involved in coding and managing homegrown asynchronous solutions



# IBM CICS Asynchronous API Concurrent Processing Made Simple

Pradeep Gohil  
Julian Horn  
Jenny He  
Anthony Papageorgiou  
Chris Poole



**Redbooks**

# Walmart and the CICS Asynchronous API

## Redbook SG24-8444

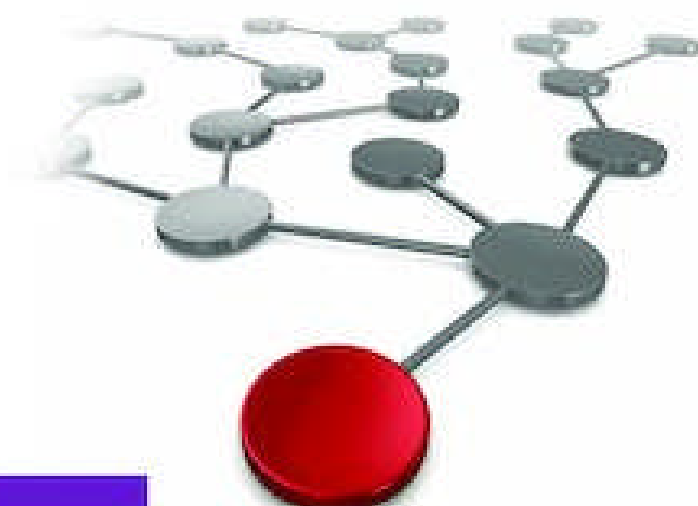
<http://www.redbooks.ibm.com/abstracts/sg248444.html?Open>

This IBM® Redbooks® publication discusses practical uses of the IBM CICS asynchronous API capability. It describes the methodology, design and thought process used by a large client, Walmart, and the considerations of the choices made. The Redbooks publication provides real life examples and application patterns that benefit from the performance and scalability offered by the new API.



## Walmart and the CICS Asynchronous API: An Adoption Experience

Pradeep Gohil  
Frank De Gales  
Nick Garrod  
Randy Fierking  
Rich Jackson  
Kathy Mathis



IBM Z

IBM

Redbooks

# Walmart and the CICS Asynchronous API

## Video course CRSE-0306

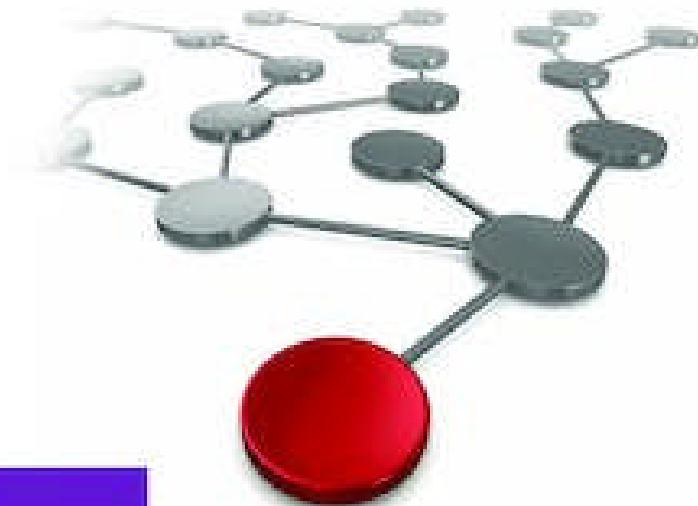
<http://www.redbooks.ibm.com/abstracts/crse0306.html?Open>

In this video series we follow the experience of the Walmart delivery team as they embrace asynchronous processing patterns in enterprise-grade applications. The series describes the methodology, design and thought process used by Walmart and the considerations of the choices made. By using IBM CICS asynchronous API, Walmart enhanced a complex search capability to achieve large scale transactions in minimal time.



## Walmart and the CICS Asynchronous API: An Adoption Experience

Pradeep Gohil  
Frank De Gales  
Nick Garrod  
Randy Fierking  
Rich Jackson  
Kathy Mathis



IBM Z

IBM

Redbooks

# Thank you

# Notices and disclaimers

Copyright © 2017-2020 by International Business Machines Corporation (IBM). No part of this document may be reproduced or transmitted in any form without written permission from IBM.

## **U.S. Government Users Restricted Rights — use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM.**

Information in these presentations (including information relating to products that have not yet been announced by IBM) has been reviewed for accuracy as of the date of initial publication and could include unintentional technical or typographical errors. IBM shall have no responsibility to update this information. **This document is distributed “as is” without any warranty, either express or implied. In no event shall IBM be liable for any damage arising from the use of this information, including but not limited to, loss of data, business interruption, loss of profit or loss of opportunity.** IBM products and services are warranted according to the terms and conditions of the agreements under which they are provided.

IBM products are manufactured from new parts or new and used parts. In some cases, a product may not be new and may have been previously installed. Regardless, our warranty terms apply.”

## **Any statements regarding IBM's future direction, intent or product plans are subject to change or withdrawal without notice.**

Performance data contained herein was generally obtained in a controlled, isolated environments. Customer examples are presented as illustrations of how those customers have used IBM products and

the results they may have achieved. Actual performance, cost, savings or other results in other operating environments may vary.

References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business.

Workshops, sessions and associated materials may have been prepared by independent session speakers, and do not necessarily reflect the views of IBM. All materials and discussions are provided for informational purposes only, and are neither intended to, nor shall constitute legal or other guidance or advice to any individual participant or their specific situation.

It is the customer's responsibility to insure its own compliance with legal requirements and to obtain advice of competent legal counsel as to the identification and interpretation of any relevant laws and regulatory requirements that may affect the customer's business and any actions the customer may need to take to comply with such laws. IBM does not provide legal advice or represent or warrant that its services or products will ensure that the customer is in compliance with any law.



## Notices and disclaimers continued

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products. IBM does not warrant the quality of any third-party products, or the ability of any such third-party products to interoperate with IBM's products. **IBM expressly disclaims all warranties, expressed or implied, including but not limited to, the implied warranties of merchantability and fitness for a particular, purpose.**

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents, copyrights, trademarks or other intellectual property right.

IBM, the IBM logo, ibm.com, Aspera®, Bluemix, Blueworks Live, CICS, Clearcase, Cognos®, DOORS®, Emptoris®, Enterprise Document Management System™, FASP®, FileNet®, Global Business Services®, Global Technology Services®, IBM ExperienceOne™, IBM SmartCloud®, IBM Social Business®, Information on Demand, ILOG, Maximo®, MQIntegrator®, MQSeries®, Netcool®, OMEGAMON, OpenPower, PureAnalytics™, PureApplication®, pureCluster™, PureCoverage®, PureData®, PureExperience®, PureFlex®, pureQuery®, pureScale®, PureSystems®, QRadar®, Rational®, Rhapsody®, Smarter Commerce®, SoDA, SPSS, Sterling Commerce®, StoredIQ, Tealeaf®, Tivoli® Trusteer®, Unica®, urban{code}®, Watson, WebSphere®, Worklight®, X-Force® and System z® Z/OS, are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at: [www.ibm.com/legal/copytrade.shtml](http://www.ibm.com/legal/copytrade.shtml).