SupportPac CA1S v1.1
CICS® TS for z/OS®: REST support in CICS using PHP

Table of Contents

This edition applies to Version 1.1 of SupportPac CA1S and to all subsequent versions, releases and modifications until otherwise indicated in new editions.

10/02/09 12:32:36

1. Introduction

1.1 What this book is about

This book explains how to use PHP with CICS® within the intended scope of this SupportPac.

1.2 What you need to know to understand this book

This book assumes that you are familiar with CICS, either as a system administrator or as a system or application programmer. It also assumes some familiarity with the PHP programming language and concepts.

1.3 Introduction to PHP on CICS

PHP is a widely-used and easy to learn scripting language. It is primarily used for writing server side Web applications. For more background information, see the Wikipedia article on PHP.

CA1S provides support for the PHP language using a Java™-based runtime package. This is a subset of the IBM® WebSphere® sMash product, which is a complete Web application platform for developing and running modern Web applications. The technology behind WebSphere sMash, including the runtime for PHP, is developed following a community driven commercial development model at www.projectzero.org.

The Java-based runtime for PHP supports almost all the language features of PHP 5.2 and a subset of the functions provided by PHP extensions. Section 13 of this document details the functionality supported by this runtime and lists known differences compared to the runtime available at php.net.

You can use the PHP scripting language capabilities in CA1S, which are shared with WebSphere sMash, to extend the range of CICS assets that can be exposed as REST services. Specifically, CA1S enables PHP programmers to:

- Write PHP scripts that are invoked in response to inbound HTTP requests.
- Access DB2® tables via PHP Data Objects (PDO) using the CICS JDBC driver from PHP code.
- Exploit existing CICS COMMAREA applications from PHP code.
- Easily create sMash-style RESTful services.

1.4 Introduction to REST

Representational state transfer (REST) is a style of software architecture that is ideally suited to exposing resources over the Web. Its key principles are:

- Application state and functionality are abstracted into resources.
- Each resource is uniquely addressable by a URI.
- A constrained, uniform set of actions is used to access and manipulate resources. This usually consists of standard HTTP methods GET, POST, PUT and DELETE.
- Requests are stateless: each request from the client must contain all the information needed for the server to understand and respond to the request.
- Requests use a pull-based interaction style: clients pull representations of state from the server.

Systems which follow these principles are often referred to as “RESTful”.

REST works particularly well with data-based services, since they are the simplest to describe in terms of resources. Furthermore, because RESTful services adhere to uniform interfaces, they facilitate the creation of "mashup" applications which mix and match data from various sources.

In the following example, the resource being exposed is "employee". The full collection of employees is represented by the URI `/employee`. Specific employees are represented by appending an identifier to the URI (e.g. `/employee/homer`). The HTTP methods define the intent of accessing the resource through a given URI:

- **URI**: `http://<host>/employee` *(the collection URI)*
    - **GET**: Returns the list of employee records
    - **POST**: Creates an employee record based on information in the body of the request.

- **URI**: `http://<host>/employee/homer` *(a member URI)*
    - **GET**: Returns the employee record "homer"
    - **PUT**: Updates the employee record "homer" based on information in the body of the request.
    - **DELETE**: Deletes the employee record "homer"

2. Installation and Configuration

2.1 Getting Started with PHP in CICS

This section gives a high level overview of the major steps required to install and set up this SupportPac in a CICS TS v3.2 system. Further details of each step are given in subsequent sections.

The main steps required to install and set up the SupportPac for PHP are:

1. Check the required prerequisites and install if necessary.
2. Unpack the SupportPac tar file to a suitable zFS location.
3. Tailor the supplied configuration files.
4. Define and install the necessary CICS resource definitions.
5. Invoke the HelloCICS sample PHP script to verify correct installation.

In the following sections more detail on each of these steps is given, together with examples which form a consistent set and if followed will provide a basic working setup.

N.B. These instructions assume that the installed CICS resources will have a prefix of CA1S, and inbound URIs for PHP will contain /ca1s in the path. If this is not the case then additional changes to the configuration files will be required during tailoring.

2.2 Prerequisites

This SupportPac is for use with CICS TS V3.2 only, and z/OS 1.8 or above.
CICS PTF PK59577 is required to allow Java 5.0 to be used in CICS.
Java SDK **5.0 SR8** must be installed and available to CICS.
CICS Web Support must be available.
CICS LOCALCCSID must be set to 037 or 1047.
TCPIP must be open in the CICS region.
ICSF must be started on the z/OS system.

For more information on configuring CICS to run Java applications, see "Java Applications in CICS" in the CICS TS 3.2 InfoCenter.

2.3 Setting up the SupportPac Files

The executable classes, native libraries, sample configuration and script files for this SupportPac are supplied as a .tar file which must be unpacked to a suitable location in zFS. The configuration files will typically need to be modified for each installation. You must correctly set up the files provided with the SupportPac to run PHP scripts in CICS.

What this SupportPac contains

This SupportPac contains a Unix tar file ca1s_v1.1.tar, which includes all of the required files in several directories.
The following directories and files are included in the SupportPac ca1s_v1.1.tar file:

ca1s/ca1s.version.id – Contains the version of the CA1S package
ca1s/licenses/* – License files for this package

ca1s/p8/ – Executables for the runtime for PHP
ca1s/p8/p8.version.id – Version of the runtime for PHP
ca1s/p8/jars/ – Jar files required for the record generator and the runtime for PHP
ca1s/p8/jars/p8api.jar – Interfaces for the runtime for PHP
ca1s/p8/jars/p8.jar – Classes for the runtime for PHP
ca1s/p8/jars/p8cics.jar – CICS-specific classes for the runtime for PHP
ca1s/p8/lib/* – Native libraries required by the runtime for PHP and extensions

ca1s/jzos/ – JZOS record generator for COBOL data structures
ca1s/jzos/jzos_recgen.jar – Jar file for the JZOS record generator for COBOL data structures
ca1s/jzos/JZOS Cobol Record Generator Users Guide.pdf – JZOS record generator user guide

ca1s/work/ – Working directory for sample setup
ca1s/work/classes/ – Default area for extra Java classes (for example, generated JZOS record classes)
ca1s/work/examples/ – Example PHP scripts for CICS in several subdirectories
ca1s/work/examples/CICS/HelloCICS.php – Installation verification script
ca1s/work/examples/CICS/info.php – Runs phpinfo()
ca1s/work/examples/CICS/logToFile.php – Example script showing how to log to file from PHP.
ca1s/work/examples/CICS/catalog/* – Example PHP scripts that interact with the 'catalog' CICS Program.
ca1s/work/examples/PDO/PDODB2Test.php – Sample PHP script using PDO to access DB2
ca1s/work/examples/REST/DataStore.php – Utility script used by sample RESTful service
ca1s/work/examples/REST/Display.php – Client for sample RESTful service
ca1s/work/examples/REST/Spaceship.php – Sample RESTful service
ca1s/work/include/ – Default area for user PHP include scripts
ca1s/work/logs/ – Working directory for JVM stdout and stderr output streams
ca1s/work/resources/ – Default area for user REST resource scripts
ca1s/work/scripts/ – Default area for regular user PHP scripts
ca1s/work/shelf/ – Working directory for CICS to copy pipeline configurations

ca1s/config/ – Configuration files for the SupportPac in several subdirectories
ca1s/config/ini/php.ini – Sample php.ini suitable for use in CICS
ca1s/config/pipelines/* – CICS PIPELINE configuration files for the sample setup
ca1s/config/profiles/CA1SJVMP – Sample JVM profile for a PHP JVM
ca1s/config/rdodefs/CA1SGRP – Sample CICS resource definition input for SupportPac setup

Unpacking the SupportPac tar file

- Transfer the tar file from your workstation to the mainframe. If you use FTP, enable binary transfer mode.
- Identify or create a suitable directory to contain the SupportPac files in the z/OS UNIX system where your CICS region runs. This directory must be accessible to the region USERID of your CICS system (with read and execute permissions). From now on, this directory will be referred to by the identifier <ca1s_loc>.
- Untar the ca1s_v1.1.tar file into the directory chosen in step 1, <ca1s_loc> (this means the directory which will contain the unpacked ca1s/ directory, rather than that directory itself). In order to avoid warning messages when the tar file is unpacked, use the –o option on the tar command e.g.
  cd <ca1s_loc>
  tar xvof <tar_loc>/ca1s_v1.1.tar
  In the above, <tar_loc> is the directory to which you copied the downloaded ca1s_v1.1.tar file.
- Check permissions for the unpacked files and directories. The CICS region userid must have read and execute access to all files under ca1s/p8, read access to files under ca1s/config and full read-write access to ca1s/work.

File encodings

All configuration files under ca1s/config/ are expected to be EBCDIC, with the exception of php.ini which may be either ASCII or EBCDIC. PHP script files are expected to be in ISO-8859-1 by default, but this can be changed by modifying the unicode.script_encoding setting in php.ini. For example, the following allows you to write PHP scripts in EBCDIC:

```
unicode.script_encoding = IBM-1047
```

When writing scripts in EBCDIC on z/OS, ensure the code page of your 3270 terminal emulator (such as IBM Personal Communications) is the same as that specified in unicode.script_encoding. If this is not the case, attempts to enter characters such as ' $' will yield unexpected behaviour, resulting in script parse errors.

The script's output is automatically converted to UTF-8 when it is sent to the client. For more information on the impact of ASCII and EBCDIC on the runtime for PHP, see the section on encoding considerations.

Editing the configuration files

As a minimum, the sample configuration files under <ca1s_loc>/ca1s/config will need to be tailored for your installation. In particular, all hard-coded instances of the directory name '/u/p8build' in the configuration files need to be replaced with <ca1s_loc>.

- The supplied php.ini file is suitable for a basic installation. The include path setting can be changed to replace the occurrence of '/u/p8build' with <ca1s_loc>. For more information on the configuration directives available in php.ini, see the php.ini reference section below.
- Edit the sample pipeline configuration files under ca1s/config/pipelines replacing '/u/p8build' with <ca1s_loc>. These files are EBCDIC encoded and should be saved as such.
- Edit the sample JVM profile ca1s/config/profiles/CA1SJVMP, replacing '/u/p8build' with <ca1s_loc>. Change CICS_HOME and JAVA_HOME to match your installation. If you plan to use DB2 from PHP, change instances of '/usr/lpp/db2910/' to match the location of your DB2 driver files.
- Edit the sample resource definitions provided in ca1s/config/rdodefs/CA1SGRP, replacing '/u/p8build' with <ca1s_loc>. Change the port number on TCPIPSERVICE CA1STCP to a suitable value. If you wish to change the prefix CA1S- used for all sample resources, you will also need to change the value of <program> in the sample pipeline configuration files to match your new name for PROGRAM CA1SHNDL.

Configuring CICS

SIT parameters

The following SIT parameters are required:

LOCALCCSID=1047 (PHP in CICS currently supports code pages 1047 or 037. If you wish to use the latter, specify LOCALCCSID=037 and change the unicode.runtime_encoding to IBM-037 in php.ini accordingly).
TCPIP=YES
JVMPROFILEDIR=… (See further comments on JVM profiles below).

Resource definitions

The supplied sample resource definitions in ca1s/config/rdodefs/CA1SGRP are in a suitable form to be used as input to a DFHCSDUP job, which will create and populate the RDO group CA1SGRP. Alternatively use the contents of ca1s/config/rdodefs/CA1SGRP as a guideline for alternative means of defining the required resources.

Once defined, install the definitions from group CA1SGRP into your running CICS system. A TCPIPSERVICE and handler PROGRAM are required. Associated URIMAPs and PIPELINE definitions may be installed in pairs depending on the use cases you wish to enable, however it is simpler to install all definitions if possible.

JVM profiles

CICS requires that JVM profiles DFHJVMPR and DFHJVMCD are always available. PHP support requires JVM profile CA1SJVMP. There are two main options:

- Set JVMPROFILEDIR=<ca1s_loc>/config/profiles and copy or create links to DFHJVMPR and DFHJVMCD in that directory.
- Copy CA1SJVMP or create a link to it in your existing JVMPROFILEDIR.

Running the supplied examples

Assuming that you installed all the sample resources as provided with the SupportPac, with only the configuration changes prescribed in the preceding sections, you are now ready to run your first PHP scripts in CICS.
Some of the supplied examples will run immediately with no further setup required, whilst others will require further steps before they can be successfully invoked. To verify correct installation and basic operation of the runtime for PHP in CICS, invoke the HelloCICS.php script from a browser by entering one of the following two URLs in your browser address field:

- http://hostname:port/ca1s/hellocics
- http://hostname:port/ca1s/cics/HelloCICS.php

In the above, hostname is the name or IP address of the system where your CICS is running, and port is the port number you chose while editing CA1SGRP or defining your TCPIPSERVICE.

The following output should appear in your browser:

```
Hello from HelloCICS.php running in CICS TS!
```

This indicates that your setup is successful and a basic installation of PHP in CICS is working correctly.

2.4 Pipeline Configuration

When an HTTP request is received, a CICS URIMAP matches the inbound URIs, thereby mapping requests to named PIPELINEs. Each PIPELINE has an xml file known as the ***pipeline configuration file***.

The pipeline configuration file can be used to influence the behaviour of the PHP engine. To do so, add an XML element **`<php_handler>`** as a child of **`<handler_parameter_list>`**. For example:

```
<?xml version="1.0" encoding="EBCDIC-CP-US"?>
<provider_pipeline xmlns="http://www.ibm.com/software/htp/cics/pipeline">
  <service>
    <terminal_handler>
      <handler>
  <program>RFPHNDLR</program>
  <handler_parameter_list>
      <php_handler prefix="/robinf" type="php_script"
               basedir="/u/robinf/scripts/" script="myscript.php" />
  </handler_parameter_list>
    </handler>
    </terminal_handler>
  </service>
</provider_pipeline>
```

The **`<php_handler>`** element has 4 optional attributes:

Attribute	Description	Default Value
`prefix`	A sub-path which will be stripped from the beginning of the inbound URI before using it to construct the path to the script. If specified, this must match a complete (i.e. up to a / boundary) prefix sub-path of the inbound URI exactly. Failure to match will result in a fatal error.	(empty string)
`type`	The type of script invocation triggered by inbound HTTP requests. Valid values are: • **`"php_script"`**: the script will be executed normally.	**`"php_script"`**

	• **"php_rest"**: the script will be invoked to handle a RESTful event.	
basedir	The base path for PHP scripts.	The home directory of the CICS region USERID.
script	The location of script to be invoked. This may be an absolute path, or a path relative to **basedir**. If unspecified, the location depends on the inbound URI and **type**: • **type="php_script"**: location is the URI with the prefix removed. If this represents a directory default file name "index.php" will be used. • **type="php_rest"**: location is the first element of the URI with prefix removed, with the suffix ".php" added if not present.	(see left)

If **type** is **php_rest**, then information about the RESTful resource targeted by an incoming request can be contained in the URI. This includes the resource name, and optionally a resource ID and further resource information.

The PHP handler uses a simple convention to determine which parts of the URI carry this information:

- The resource name is the first URI element after the **prefix**.
- The resource id is the second URI element after the **prefix**.
- Any further URI elements up to the query string represent additional resource information.

http://xyz.com/<…pre/fix…>/<resourceName>/<resourceID>/<…more/resource/info…>

2.5 Pipeline Configuration Examples

No pipeline configuration, or <php_handler />

Inbound URI	Target Script
http://xyz.com/a.php	$HOME/a.php
http://xyz.com/b/c.php	$HOME/b/c.php
http://xyz.com/b	$HOME/b/index.php

<php_handler basedir="/u/robinf/scripts" />

Inbound URI	Target Script
http://xyz.com/a.php	/u/robinf/scripts/a.php
http://xyz.com/b/c.php	/u/robinf/scripts/b/c.php
http://xyz.com/b	/u/robinf/scripts/b/index.php

<php_handler basedir="/u/robinf/scripts" script="foo.php" />

Inbound URI	Target Script
http://xyz.com/a.php	/u/robinf/scripts/foo.php
http://xyz.com/b/c.php	/u/robinf/scripts/foo.php
http://xyz.com/b	/u/robinf/scripts/foo.php

<php_handler prefix="/team/forum" basedir="/u/robinf/forum2.0" />

Inbound URI	Target Script
http://xyz.com/team/forum	/u/robinf/forum2.0/index.php
http://xyz.com/team/forum/post.php	/u/robinf/forum2.0/post.php
http://xyz.com/team/forum/db/config.php	/u/robinf/forum2.0/db/config.php

```
<php_handler type="php_rest"
      prefix="/my/service" basedir="/u/robinf/restlib" />
```

Inbound URI	Target Script
`http://xyz.com/my/service/photo`	`/u/robinf/restlib/photo.php` *(resource name: "photo")*
`http://xyz.com/my/service/user`	`/u/robinf/restlib/user.php` *(resource name: "user")*
`http://xyz.com/my/service/user/303`	`/u/robinf/restlib/user.php` *(resource name: "user",* *resource ID: "303")*
`http://xyz.com/my/service/user/303/address/postcode`	`/u/robinf/restlib/user.php` *(resource name: "user",* *resource ID: "303",* *resource info: "/address/postcode"*)

```
<php_handler type="php_rest"
      prefix="/my/service" basedir="/u/robinf/restlib"
      script="entrypoint.php" />
```

Inbound URI	Target Script
`http://xyz.com/my/service/photo`	`/u/robinf/restlib/entrypoint.php` *(resource name: "photo")*
`http://xyz.com/my/service/user`	`/u/robinf/restlib/entrypoint.php` *(resource name: "user")*
`http://xyz.com/my/service/user/303`	`/u/robinf/restlib/entrypoint.php` *(resource name: "user",* *resource ID: "303")*
`http://xyz.com/my/service/user/303/address/postcode`	`/u/robinf/restlib/entrypoint.php` *(resource name: "user",* *resource ID: "303",* *resource info: "/address/postcode"*)

3. Accessing Request Information with zget() and zlist()

3.1 Overview

Functions **`zget()`** and **`zlist()`** emulate a small part of WebSphere sMash's Global Context feature. Together, they provide easy access to information about the current request and RESTful event from within a PHP script.

These functions are part of the CICS extension, which is enabled in php.ini with the following entry:

```
      extension = com.ibm.p8.library.cics.CICSLibrary
```

The extension is enabled in the php.ini shipped in the package.

- **`zget($key [, $defaultValue=NULL])`** returns the value corresponding to key **`$key`**, or **`$defaultValue`** if no such key exists. Valid keys are:

Key	Type	Description
/request/method	string	The HTTP method (GET, POST, PUT, DELETE)
/request/input	stream	Stream providing access to the raw bytes of the request content. Use this for accessing non-form-encoded post or put data
/request/input/transcoded	string	The request input data transcoded into a string using the runtime encoding. This is specific to the CICS extension and is not available on sMash.
/request/params/<name>	string	The value of form element <name>. Includes query

		string elements, as well as POST or PUT data **if they are form encoded** (application/x-www-form-urlencoded). Input that is not form-encoded can be accessed through /request/input and /request/input/transcoded
/request/params/<resourceName>Id	string	The id of the resource if this is a RESTful request to member URI. See sMash docs for more info.
/event/pathInfo	string	The part of the URI following the resource ID if this is a RESTful request. See sMash docs for more info.
/event/_name	string	The name of the event if this is a RESTful request (list, create, retrieve, update, delete, postMember, putCollection or deleteCollection). See sMash docs for more info. Note that CICS also supports automatic dispatch to the appropriate PHP method for the event – see below.

- **`zlist($prefix [, $includePrefix=false])`** returns the list of keys starting with $prefix. The optional argument $includePrefix defines whether or not the prefix should be included in the result.

For example, assuming the request includes form data **`a=1&b=2&c=3`**:

```
<?php
  foreach (zlist("/request/params", false) as $key) {
    echo "$key: " . zget("/request/params/" . $key) ."\n";
  }
  foreach (zlist("/request/params", true) as $key) {
    echo "$key: " . zget($key) ."\n";
  }
?>
```

Output:

```
  a: 1
  b: 2
  c: 3
  /request/params/a: 1
  /request/params/b: 2
  /request/params/c: 3
```

3.2 Limitations and Differences Compared to IBM WebSphere sMash

- On sMash, the global context is read/write. On CICS, it is read-only (there is no zput(), zpost...).
- On sMash, many keys are available. On CICS, only the keys listed above are available. Note: if you need information about the HTTP request that is not available through zget on CICS, you might find it in the $_SERVER superglobal.
- On sMash, different zones of the Global Context have different lifetimes: some entries survive for only a part of a request, while others survive across multiple requests. On CICS, everything that can be accessed via **`zget()`** or **`zlist()`** is valid for the duration of the current request: all available keys are from the /request and /event zones, and it is not possible to fire multiple events per request.
- CICS provides **`zget("/request/input/transcoded")`** which returns non-form-encoded input as a string, transcoded to the runtime encoding. This is not applicable to sMash.

- On CICS, no items accessible with **`zget()`** are Lists, so the syntax for directly accessing List and FirstElementList elements is **not** available.

4. RESTful Events and Event Handlers

4.1 RESTful Events

When the PHP handler is in REST mode (**`type="php_rest"`** in the pipeline configuration), any inbound HTTP request corresponds to one of 8 possible RESTful event types. The event type is determined by two factors:

- the HTTP method of the request
- whether the request URI corresponds to a collection or a specific member – that is, whether a resource ID is specified in the URI

Request URI	HTTP Method	Event
Collection URI, e.g.: `http://xyz.com/prefix/`**`myResource`**	GET	**list**
	POST	**create**
	PUT	**putCollection**
	DELETE	**deleteCollection**
Member URI, e.g.: `http://xyz.com/prefix/`**`myResource/resourceID`**	GET	**retrieve**
	POST	**postMember**
	PUT	**update**
	DELETE	**delete**

The event type can be determined programmatically within the script using **`zget('/event/_name')`**. This can be used in conjunction with **`zget('/request/<resourceName>Id')`** and **`zget('/event/pathInfo')`** to easily construct scripts to respond to RESTful requests.

For example, the following script would respond as expected to GET requests to URIs **`http://xyz.com/resources/user`**, **`http://xyz.com/resources/user/123`** and **`http://xyz.com/resources/user/123/address`**.

user.php

```php
<?php
switch(zget('/event/_name')) {
  case 'list':
      // list all users (/user)
    break;
  case 'retrieve':
      $userId = zget('/request/params/userId');
      $info = zget('/event/pathInfo');
      if ($info == null) {
        // show all info for user $userId (/user/123)
      } else {
        // show $info for user $userId (/user/123/address)
      }
    break;
  default:
      // Do nothing for other event types
}
?>
```

4.2 Event Handler Methods

Developers may also choose a more object-oriented approach to event handling:

- After executing the script, the PHP engine searches for a class with the same name as the resource.
- If such a class is defined, an instance is created. The constructor is invoked if it is present and does not require arguments.
- Finally, a method corresponding to the event is invoked, if present:

Request URI	HTTP Method	Event Handler
`http://xyz.com/prefix/`**`user`**	GET	**User::onList()**
	POST	**User::onCreate()**
	PUT	**User::onPutCollection()**
	DELETE	**User::onDeleteCollection()**
`http://xyz.com/prefix/`**`user/123`**	GET	**User::onRetrieve()**
	POST	**User::onPostMember()**
	PUT	**User::onUpdate()**
	DELETE	**User::onDelete()**

In other words, the following script is equivalent to the script above:

user.php
```
<?php
class User {
  function onList() {
      // list all users (/user)
  }

  function onRetrieve() {
      $userId = zget('/request/params/userId');
      $info = zget('/event/pathInfo');
      if ($info == null) {
        // show all info for user $userId (/user/123)
      } else {
        // show $info for user $userId (/user/123/address)
      }
  }
}
?>
```

Notes:

- Services may implement as few or as many handlers as necessary – there is no requirement to implement them all. The most commonly used events are those that represent "LCRUD" operations: onList, onCreate, onRetrieve, onUpdate and onDelete.

- The outer scope script is always invoked prior to the method dispatch. This means that any code outside the class definition (including code in any included files) will be executed for all operations. Therefore, only common code applicable to all operations should be placed outside the class definition.

- The class is instantiated prior to method dispatch. If a constructor which takes no arguments is present, then it is called.

- The class does not need to be defined in the PHP file specified in the HTTP request: it can be defined in an **`include`**d PHP script.

- In PHP, class names and function names are case-insensitive. So whilst it is good practice to use names such as **`onList()`**, in fact **`onlist()`** would also match.

5. An Example RESTful Service

A simple RESTful service is included in the SupportPac. It demonstrates the usage of **`zget()`** and the event handler mechanism.

5.1 Overview

The resource used in the example is ***spaceship***, which has 2 attributes: “x” and “y”. Both are non-zero positive integers.

Spaceship.php provides handlers for LCRUD operations on spaceships. All output data returned by the service is in JSON format. In order to illustrate reading both form-encoded data and non-form-encoded data, the service expects form input data (with Content-Type: “application/x-www-form-urlencoded”) for Create operations and JSON input data for Update operations.

Here are some operations supported by the service:

HTTP Method	Request URI	Result
GET	`/Spaceship`	Returns HTTP 200 with JSON list of all spaceships, with their x and y values.
	`/Spaceship?x=1`	As above, but filtered to include only ships where x=1.
	`/Spaceship/1`	Returns HTTP 200 with JSON representation of Spaceship 1’s x and y values, or HTTP 404 if no such ship exists
	`/Spaceship/1/x`	Returns HTTP 200 with JSON representation of Spaceship 1’s x value, or HTTP 404 if no such ship exists
POST	`/Spaceship`	Create spaceship, assuming requests include form-encoded data like: `id=3&x=2&y=4`. Returns HTTP 200 with JSON representation of the newly created ship, or HTTP 400 if the input data is invalid, or HTTP 409 if supplied id matches an existing ship.
PUT	`/Spaceship/1`	Update spaceship 1 with JSON data supplied in the request. Returns HTTP 200 with the representation of the ship prior to the update, or HTTP 201 if the ship did not exist and was created, or HTTP 400 if the input data is invalid.
	`/Spaceship/1/x`	Update attribute x of spaceship 1 with JSON data supplied in the request. Returns HTTP 200 with the representation of the ship prior to the update or HTTP 400 if the input data is invalid.
DELETE	`/Spaceship/1`	Deletes spaceship 1. Returns HTTP 200 with JSON representation of the deleted ship, or HTTP 404 if no such ship exists.

DataStore.php implements the persistence mechanism for the service. For simplicity, each ship is represented by a separate file containing the ship’s “x” and “y” attributes.

Display.php is a simple client which issues regular asynchronous GET requests to the collection URI, and accordingly adjusts HTML elements representing ships 1 to 9.

5.2 How to Use the Example Service

- The script files that constitute the example are supplied under **`work/examples/REST`**. In a default CA1S setup, the Spaceship service and Display page are available under the following URIs:
  ```
  http://<host>:<port>/ca1s/rest/Spaceship
  http://<host>:<port>/ca1s/rest/Display
  ```
  This uses the URIMAP and PIPELINE named **`CA1SREST`** and the config file **`phppipe_REST.xml`**.

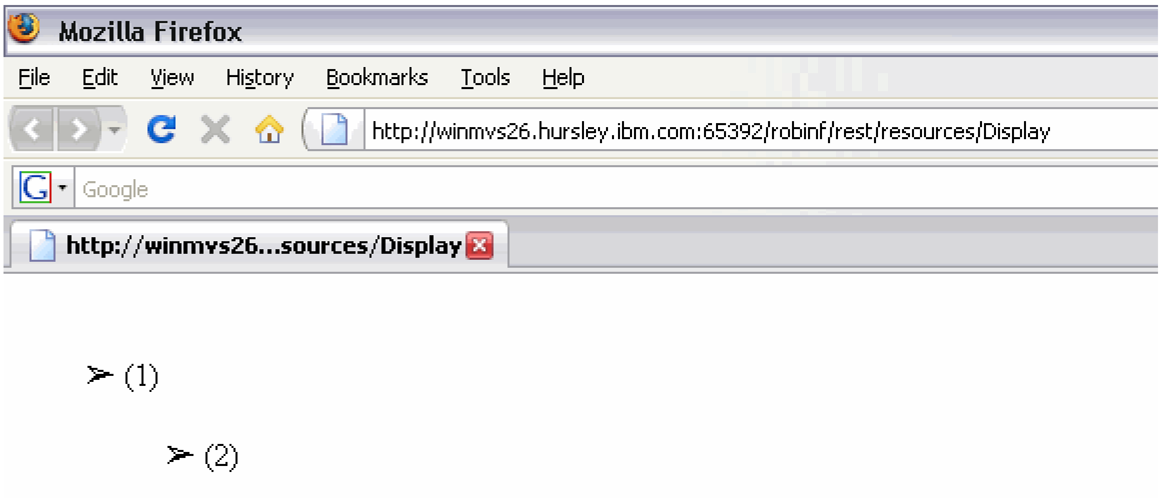
T

- Optionally,you may create your own URIMAP and PIPELINE to customize the location of the script files or the URI of the service. You will need a RESTful pipeline config as described in section 2.4 and to ensure files **Spaceship.php**, **Display.php** and **DataStore.php** are copied to the appropriate resource directory.
  For example, with this pipeline configuration:

  ```
  <php_handler type='php_rest' prefix='robinf/rest/resources'
               basedir='/u/robinf/cicswork/scripts/robinf/restlib/' />
  ```

  the scripts would be copied into **`u/robinf/cicswork/scripts/robinf/restlib/`** and accessed using URIs like **`http://<host>:<port>/robinf/rest/resources/Spaceship`** .

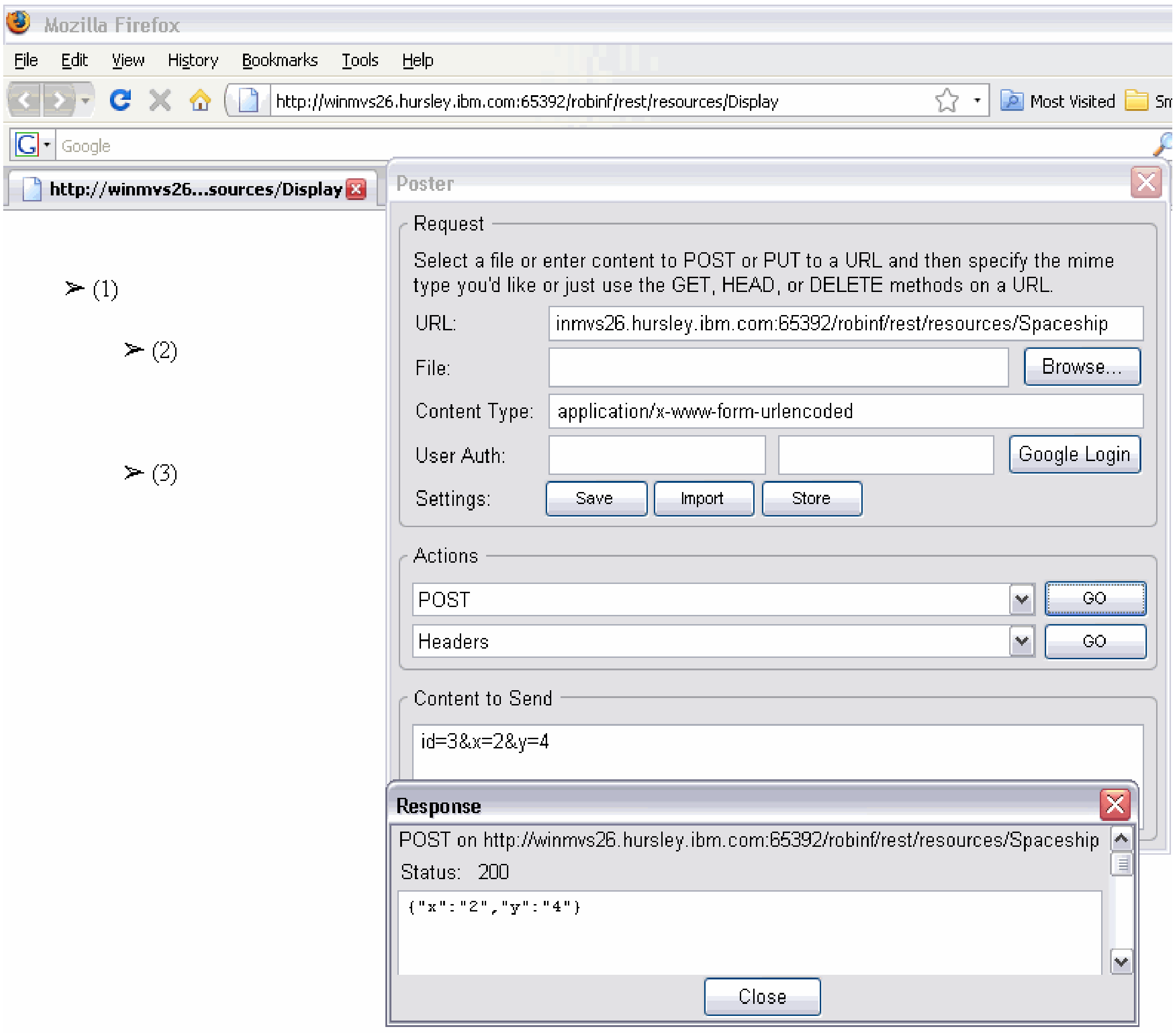
- Browse to /Display. If there is no data in the data store (as will be the case initially), the application will populate the data store with two ships. The page will then update to display them:


Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://winmvs26.hursley.ibm.com:65392/robinf/rest/resources/Display
Google
http://winmvs26...sources/Display

➢ (1)

➢ (2)

- To add a ship, issue a POST request to /Spaceship with form-encoded data like `id=3&x=2&y=4`. Ensure that the request specifies "application/x-www-form-urlencoded" as the Content-Type. In the screenshot below, the Poster add-on for Firefox® is used as a simple REST client to interact with the service:



Mozilla Firefox

http://winmvs26.hursley.ibm.com:65392/robinf/rest/resources/Display

➢ (1)

➢ (2)

➢ (3)

Poster

Request

Select a file or enter content to POST or PUT to a URL and then specify the mime type you'd like or just use the GET, HEAD, or DELETE methods on a URL.

URL: inmvs26.hursley.ibm.com:65392/robinf/rest/resources/Spaceship

File:

Content Type: application/x-www-form-urlencoded

User Auth:

Actions

POST

Headers

Content to Send

```
id=3&x=2&y=4
```

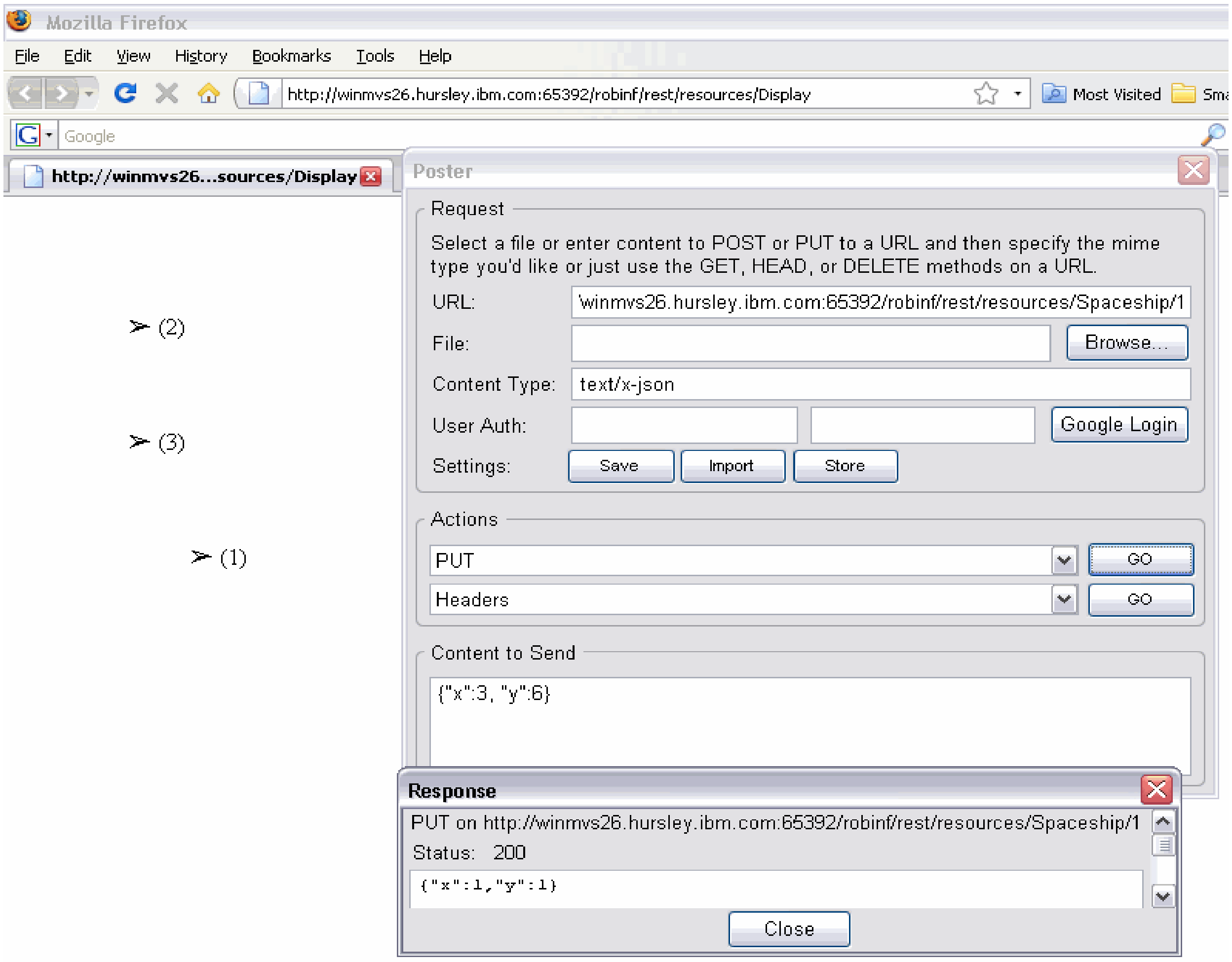
Response

POST on http://winmvs26.hursley.ibm.com:65392/robinf/rest/resources/Spaceship

Status: 200

```
{"x":"2","y":"4"}
```

- To update a ship's attributes, issue a PUT request to /Spaceship/<id> with JSON like `{"x ":1, "y ":2} :`



Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://winmvs26.hursley.ibm.com:65392/robinf/rest/resources/Display
Most Visited
Google
http://winmvs26...sources/Display
➢ (2)
➢ (3)
➢ (1)
Poster
Save Import Store
Browse...
Google Login
GO
GO
Close

Request

Select a file or enter content to POST or PUT to a URL and then specify the mime type you'd like or just use the GET, HEAD, or DELETE methods on a URL.

URL: winmvs26.hursley.ibm.com:65392/robinf/rest/resources/Spaceship/1

File:

Content Type: text/x-json

User Auth:

Settings:

Actions

PUT

Headers

Content to Send

{"x":3, "y":6}

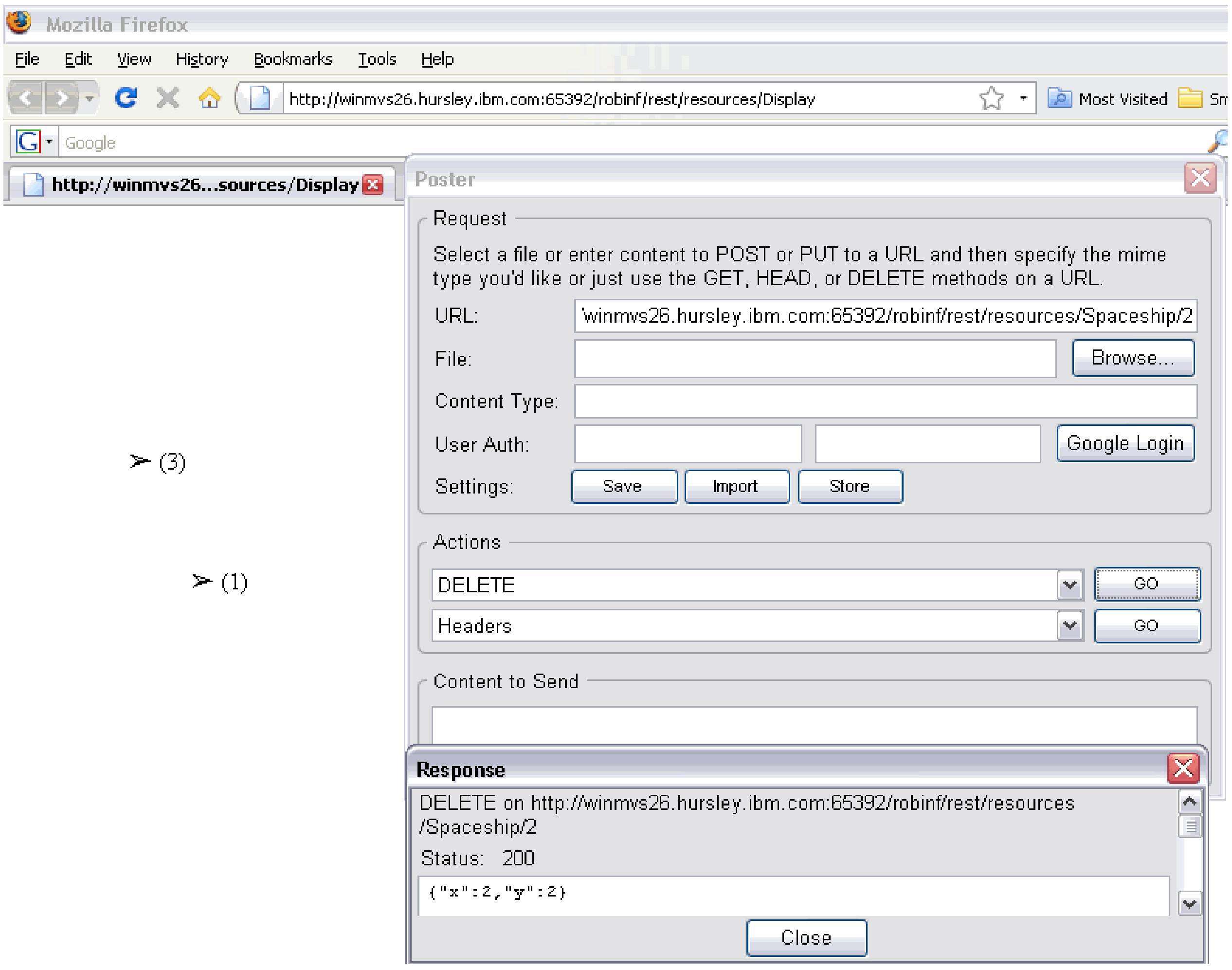
Response

PUT on http://winmvs26.hursley.ibm.com:65392/robinf/rest/resources/Spaceship/1

Status: 200

```
{"x":1,"y":1}
```

- To delete a ship's attributes, issue a DELETE request to /Spaceship/<id> :


Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://winmvs26.hursley.ibm.com:65392/robinf/rest/resources/Display
Most Visited
Google
http://winmvs26...sources/Display
➢ (3)
➢ (1)
Poster
Browse...
Google Login
Save
Import
Store
GO
GO
Close

Request

Select a file or enter content to POST or PUT to a URL and then specify the mime type you'd like or just use the GET, HEAD, or DELETE methods on a URL.

URL: winmvs26.hursley.ibm.com:65392/robinf/rest/resources/Spaceship/2

File:

Content Type:

User Auth:

Settings:

Actions

DELETE

Headers

Content to Send

Response

DELETE on http://winmvs26.hursley.ibm.com:65392/robinf/rest/resources /Spaceship/2

Status: 200

```
{"x":2,"y":2}
```

6. Calling CICS Programs from PHP

The CICS extension allows a PHP programmer to call a CICS program from a PHP script. The extension currently supports CICS programs that can be called using EXEC LINK, and that communicate using a commarea data structure.

6.1 Using the CICS Extension to Call a CICS Program

Typical Work flow

1. From a COBOL program, use the COBOL compiler to generate a binary 'adata' file, which contains meta-data describing a COBOL copybook from the program.
2. Use supplied JZOS tooling to parse the 'adata' file and produce a Java source file.
3. Compile the resulting source to Java class files. These classes allows the creation of a commarea data structure that can be passed to a CICS program.
4. Ensure the generated classes are on the Java CLASSPATH.
5. Use PHP reflection to determine the available methods on the generated Java class.
6. Write a PHP program that:
   - Uses the Java bridge to access the generated JZOS classes (no knowledge of the Java programming language required).

- Uses these classes to generate an appropriate commarea
- Uses the CICS extension to call a CICS program (passing the generated commarea)
- Examines the result using bridged JZOS classes.

In order to successfully use the CICS extension for PHP:

- The CICS extension must be specified in the php.ini file:
  `extension = com.ibm.p8.library.cics.CICSLibrary`
- The generated classes representing the commarea of the CICS program to be called must be available on the Java CLASSPATH. The Java CLASSPATH can be set in the JVM profile using the `CLASSPATH_SUFFIX` setting. If you are using the supplied JVM profile (CA1SJVMP), you can use the directory `ca1s/work/classes/`, which already is set up to be on the CLASSPATH.

Example Scripts

A simplified example script is show below:

```
<?php
// Use the Java bridge to import the commarea class.
// Call java_import specifying the full package name (if any)
// and the class name, but not the .class extension.
java_import("package.name.GeneratedJzosClass");

// Instantiate the class as a PHP class
$commArea = new GeneratedJzosClass();

// Set some data in the commarea by calling method on the class
$commArea->setCallType("1");


// Use the CICS extension to call a CICS program
$program = new CICSProgram("ProgramName");
try {
      $program->link($commArea);
} catch (CICSException $e) {
      echo $e->getMessage();
      exit;
}

echo  "Return value is " . $commArea->getReturnValue();
?>
```

An example of determining available methods by PHP reflection:

```
<?php

// Use reflection to get an array of methods available on a
// generated commarea class
$generatedClass = new ReflectionClass("GeneratedJzosClass");
$methods = $generatedClass->getMethods();
foreach ($methods as $method) {
      // Print each method name
      echo $method->getName() . "\n";
}

?>
```

6.2 Representing Commarea Data Structures in PHP

Commarea data structures are represented by generated Java classes. The Java source code for these classes is generated by the JZOS record generator tooling, using binary ADATA files created by the COBOL compiler. The JZOS record generator tool and its documentation are included in the package under ca1s/jzos. More information on JZOS, along with updates, FAQs and a forum, are available on the IBM JZOS Batch Toolkit for z/OS SDKs site at alphaworks.

Generating ADATA files

Use the COBOL compiler on the program containing the data structure of interest. Use the option ADATA and specify a DD card SYSADATA to indicate where the adata is to be stored. See "Enterprise COBOL for z/OS Programming Guide" for more details.

If you use FTP to transfer the ADATA to your workstation, remember to issue the command '**quote site rdw**'. If this step is omitted, the record generation step will fail with a java.lang.NegativeArraySizeException or similar. For more information, see section "Running the COBOL RecordClassGenerator" of "JZOS Cobol Record Generator Users Guide.pdf".

Generating the Java source files using JZOS

Detailed documentation for generating these classes is available in the “JZOS Cobol Record Generator Users Guide.pdf” file in the ca1s/jzos directory. A simple example invocation of the generator is:

```
java -cp jzos_recgen.jar com.ibm.jzos.recordgen.cobol.RecordClassGenerator
genCache=false adataFile=SOMEADATA symbol=DSECTname class=GENCLASS
package=sample
```

Note that the above is a single command and should be pasted as one line.

Parameters:

- adataFile – the file containing the binary adata output from the COBOL compiler.
- class – the class name for the generated Java record class (optional: if omitted, a default name will be generated).
- genCache=false – disable caching in the generated Java source files. **NB: This parameter is required: genCache must be set to false for correct interaction between PHP and COBOL.**
- package – the Java package name to be used for the generated Java class (optional). The complete generated class name used to refer to the commarea class in PHP is formed by concatenating “package.class”.
- symbol – the name of the level 01 data description entry in the COBOL program that describes the data structure of interest. This is required unless the COBOL source program contains only one level 01 data description entry.

6.3 API Documentation

CICSProgram Class

`CICSProgram->__construct(string $programName)`

Description:
Create a new CICSProgram object

Parameters:

- **`programName:`** The name of the CICS program to call.

`boolean CICSProgram->link([object $inputCommArea [, object $outputCommArea]])`

Description:
Call the CICS program

Parameters:
- **`inputCommArea`**: a commarea that will be passed to the CICS program as input data. This object should be an imported Java class that has a public method with signature: byte[] getByteBuffer() (such as generated by the JZOS tooling). If no output commarea is supplied, the results of calling the CICS program will be overwritten into this object.
- **`outputCommArea`**: a commarea that will have the output from the CICS program written into it. Again this should be an imported Java class generated by the JZOS tooling.

Return Values:
- Returns true if the link succeeded, otherwise false.

Errors/Exceptions:
- Produces an E_WARNING for any errors detected in the input data, and returns false.
- Throws a CICSException if an error occurs during execution of the CICS program.

`boolean CICSProgram->link_sync([object $inputCommArea [, object $outputCommArea]])`

Description:
Call the CICS program. If the program is remote, a SYNCPOINT will be performed when the program returns.

Parameters:
- **`inputCommArea`**: a commarea that will be passed to the CICS program as input data. This object should be an imported Java class that has a public method with signature: byte[] getByteBuffer() (such as generated by the JZOS tooling). If no output commarea is supplied, the results of calling the CICS program will be overwritten into this object.
- **`outputCommArea`**: a commarea that will have the output from the CICS program written into it. Again this should be an imported Java class generated by the JZOS tooling.

Return Values:
- Returns true if the link succeeded, otherwise false.

Errors/Exceptions:
- Produces an E_WARNING for any errors detected in the input data, and returns false.
- Throws a CICSException if an error occurs during execution of the CICS program.

CICSException Class

`object CICSException->getCause()`

Description:
get the underlying Java exception that caused this `CICSException`

Return Values:
The underlying Java exception if one exists, or null otherwise.

The following example shows how to display the abend code included in an underlying Java AbendError:

```
try {
      $program->link($commArea);
} catch (CICSException $e) {
      echo $e->getCause()->getABCODE();
      exit;
}
```

6.4 Sample Application

To illustrate these features, a simple PHP front-end to the CICS catalog example program is supplied under `ca1s/work/examples/CICS/catalog/`. You can examine the code to learn how PHP scripts interact with CICS programs.

To use the sample:

- Ensure the catalog CICS program is installed and available to your CICS environment. It must be named `DFH0XCMN`. For more information on setting up the catalog CICS program, please refer to chapter 14 of the Web Services Guide in the CICS TS 3.2 InfoCenter.

- Add `ca1s/work/examples/jars/catalog.jar` to the Java CLASSPATH by appending it to CLASSPATH_SUFFIX in your JVMProfile. This jar file contains the catalog commarea classes that were generated from the catalog ADATA using JZOS.

- **Optional:** Alternatively, instead of using the supplied jar, you could generate and compile the classes from the catalog program's ADATA yourself, specifying 'sample' as the package name. For example, generate the source using the following command (all on one line):

  ```
  java -cp jzos_recgen.jar com.ibm.jzos.recordgen.cobol.RecordClassGenerator
  genCache=false adataFile=DFH0XCMN symbol=DFHCOMMAREA class=DFH0XCMN
  package=sample outputDir=.
  ```

  This will create a directory named `sample` containing a Java source file. Compile it like this:

  ```
  javac -cp jzos_recgen.jar sample/*
  ```

  The resulting `sample` directory, which will now include 2 .class files, should be placed inside a directory that is in CLASSPATH_SUFFIX in your JVMProfile. If you are using the supplied JVMProfile (CA1SJVMP), you can use the directory `ca1s/work/classes/`, which already is set up to be on the CLASSPATH.

- Phase out your JVMs to ensure the JVMProfile change is picked up.

- Navigate to catalog.php in your browser. With the default setup, this will be located at: `http://<host>:<port>/ca1s/cics/catalog/catalog.php`

7. Accessing DB2® Databases from PHP

7.1 Getting Started

The following steps describe how to access DB2 databases from PHP using the PDO extension:

1. Edit **php.ini** and locate the section where the PHP extensions are configured. Uncomment the following two lines:

   ```
   extension = com.ibm.p8.library.pdo.PdoLibrary
   extension = com.ibm.p8.library.pdo.PdoJdbcDb2Library
   ```

   These lines ensure the PDO extension and the PDO/DB2 driver are loaded into the runtime for PHP.

2. Ensure the following DB2 runtime JARs are on the Java CLASSPATH by adding them to CLASSPATH_SUFFIX in the CA1SJVMP JVMProfile:

   ```
   db2jcc.jar
   ```

```
db2jcc_javax.jar
db2jcc_license_cisuz.jar
```

These can normally be found in the **classes** subdirectory of the DB2 installation.

3. A suitable DB2 instance must be available to CICS, and an active DB2 connection must be installed and connected within the CICS region.

4. Copy and save the following sample code into a PHP script available to the CICS environment:

```
<?php

$db = new PDO("jdbc:default:connection");
if ($db == FALSE) {
     die("Could not create PDO object!");
}

$db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_WARNING);
$db->setAttribute(PDO::ATTR_CASE, PDO::CASE_LOWER);
$db->setAttribute(PDO::ATTR_STRINGIFY_FETCHES, true);

$tablename = "PDODB2Test";

$savedErrorMode = $db->getAttribute(PDO::ATTR_ERRMODE);
$db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_SILENT);
$db->exec("DROP TABLE $tablename");
$db->setAttribute(PDO::ATTR_ERRMODE, $savedErrorMode);

$db->exec("CREATE TABLE $tablename(id int NOT NULL PRIMARY KEY, val
VARCHAR(10))");
$db->exec("INSERT INTO $tablename VALUES(1, 'A')");
$db->exec("INSERT INTO $tablename VALUES(2, 'B')");
$db->exec("INSERT INTO $tablename VALUES(3, 'C')");

$stmt = $db->prepare("SELECT * from $tablename ORDER BY id");
$stmt->execute();
var_dump($stmt->fetchAll(PDO::FETCH_ASSOC));

?>
```

5. Point a web browser at the PHP script.

6. The following output should be displayed:

```
array(3) {
  [0]=>
  array(2) {
  ["id"]=>
  string(1) "1"
  ["val"]=>
  string(1) "A"
  }
  [1]=>
  array(2) {
  ["id"]=>
  string(1) "2"
  ["val"]=>
  string(1) "B"
  }
```

```
   [2]=>
   array(2) {
   ["id"]=>
   string(1) "3"
   ["val"]=>
   string(1) "C"
   }
 }
```

7.2 Supported Functions

Method	Currently Supported?		Notes (see below)
	Yes	No	
PDO::beginTransaction		✘	
PDO::commit		✘	
PDO::__construct	✓		1
PDO::errorCode		✘	2
PDO::errorInfo		✘	2
PDO::exec	✓		
PDO::getAttribute	✓		3
PDO::getAvailableDrivers	✓		4
PDO::lastInsertId	✓		
PDO::prepare	✓		5, 7
PDO::query	✓		
PDO::quote	✓		
PDO::rollBack		✘	
PDO::setAttribute	✓		3
PDOStatement::bindColumn		✘	
PDOStatement::bindParam		✘	
PDOStatement::bindValue	✓		5
PDOStatement::closeCursor	✓		
PDOStatement::columnCount		✘	
PDOStatement::errorCode		✘	2
PDOStatement::errorInfo		✘	2
PDOStatement::execute	✓		5
PDOStatement::fetch	✓		6
PDOStatement::fetchAll	✓		6
PDOStatement::fetchColumn		✘	
PDOStatement::fetchObject		✘	
PDOStatement::getColumnMeta		✘	
PDOStatement::nextRowset		✘	
PDOStatement::rowCount	✓		
PDOStatement::setAttribute		✘	
PDOStatement::setFetchMode		✘	

Notes:

1. The format of the Data Source Name provided to the constructor is the same as that required by the underlying JDBC driver being used.

2. The only supported manner to programatically interrogate and recover from database-originated errors is to set the attribute PDO::ATTR_ERRMODE to PDO::ERRMODE_EXCEPTION, using the method PDO::setAttribute, which will cause a PDOException to be thrown when any database-originated error is found. This exception may then be caught and handled in the standard manner.

3. The currently supported attributes on the PDO object are PDO::ATTR_CASE and PDO::ATTR_ERRMODE. In addition, PDO::ATTR_STRINGIFY_FETCHES is handled without error, but not currently honoured.

4. The list of drivers returned reflect the PDO driver-specific (Java) extension libraries that have been loaded - they do not necessarily reflect the existence of the supporting JDBC driver libraries required to use the driver successfully.

5. Positional SQL IN parameters are supported; named parameters, IN/OUT and OUT parameters are currently unsupported. Date fields must be set by a string of the form "yyyy-MM-dd"; Time fields by a string of the form "HH:mm:ss"; and Timestamp fields by a string of the form "yyyy-MM-dd HH:mm:ss". The setting of parameters for database fields mapping to the following JDBC datatypes (java.sql.Types) are not currently supported: ARRAY, BLOB, CLOB, DATALINK, JAVA_OBJECT, OTHER, REF, STRUCT.

6. The following fetch styles are supported: PDO::FETCH_ASSOC, PDO::FETCH_NUM and PDO::FETCH_BOTH. In addition, the following fetch style modifiers are supported on calls to PDOStatement::fetchAll: PDO::FETCH_UNIQUE and PDO::FETCH_GROUP. All other fetch styles are not currently supported.

7. The syntax for standard and precompiled SQL statements (i.e. JDBC PreparedStatements) is supported; the syntax for SQL stored procedures (i.e. JDBC CallableStatements) is not currently supported.

8. Managing Units of Work

8.1 Explicitly Committing and Rolling Back Units of Work from PHP

PHP scripts can commit and roll back units of works using the class `CICSTask`, which provides two static methods: `CICSTask::commit()` and `CICSTask::rollback()`. This class cannot be instantiated.

```
class CICSTask {
      public static function commit()
      public static function rollback()
}
```

`CICSTask::commit()`

- Commit the work done as part of the request.
- **Arguments:** None
- **Return value:** NULL
- **Errors conditions:** throws `CICSException` if the commit request could not be successfully completed and the current Unit of Work has been rolled back.

`CICSTask::rollback()`

- Roll back the work done as part of the request.
- **Arguments:** None

- **Return value:** NULL

An example usage scenario:

```
<?php

do_some_work();

if (!verify_state()) {
  CICSTask::rollback();
  return;
}

try {
  CICSTask::commit();
} catch (CICSException $e) {
  // Commit failed - LUW has been rolled back! Handle as appropriate.
  do_severe_issue_handling($e);
  return;
}
?>
```

8.2 *Implicit Commits and Rollbacks*

A commit is implicitly issued after a script ends without failure.

A rollback is implicitly issued in the following cases:

If the PHP engine fails during the execution of the script, for example because of a JVM crash.

If an error in PHP code causes the script execution to halt, for example because a PHP fatal error is triggered, or a PHP exception is thrown and not caught.

Note: only errors that halt the script trigger a rollback. Therefore, Warnings and Notices do not trigger a rollback. See the php.net manual entry on error types for more information on PHP errors.

9. Encoding Considerations

EBCDIC, which stands for Extended Binary-Coded-Decimal Interchange Code, is the code set used on z/OS UNIX. By contrast, ASCII is the dominant code set on almost all other platforms. Scripts written for ASCII platforms without an awareness of code set independent coding practices can run incorrectly on z/OS UNIX. The following sections discuss some of these issues.

For a general overview of character sets and code pages, refer to *National Language Support Reference Manual, Volume 2*, SE09-8002.

9.1 *PHP Script File Encoding and php.ini Directives*

There are two directives in php.ini that relate to encoding:

- **unicode.script_encoding**: Script encoding describes the encoding of the PHP script files to be read. This must be correctly set in order for string literals and PHP names, such as function, variable and class names, to be correctly parsed by the runtime. The default is UTF-8. If this is set incorrectly, the script will not parse.

- **unicode.runtime_encoding**: Runtime encoding is used when a PHP string is converted into a Java string. Internally, the runtime preserves the ability to store binary data in a PHP string, which is essential for full support of the PHP 5 language. This setting must match LOCALCCSID (which will typically be either 1047 or 037), except it must include the prefix “IBM-” (i.e. “IBM-1047” or “IBM-037”). Due to changes in the iconv library, on z/OS v1.9 it is acceptable to specify the code page name in php.ini without a dash, i.e. “IBM037” or “IBM1047”. However, note that on z/OS v1.8, the dash is required: “IBM-037” or “IBM-1047”. Script output will automatically be transcoded by CICS from the runtime encoding to UTF-8 when it is returned to the client.

9.2 Transferring PHP Scripts over FTP

When transferring scripts over FTP, ensure the correct transfer mode is enabled:

- Binary mode will preserve a script's encoding when it is transferred between systems. This is useful for keeping scripts in ASCII on both the z/OS system and your workstation.
- Text mode may convert the script between ASCII and EBCDIC depending on the source and destination systems. This can be used to edit scripts on your workstation in ASCII, then deploy them to z/OS in EBCDIC.

9.3 Sort Order Differences

The default sort order for text differs between ASCII and EBCDIC platforms. When using the PHP sort() function, you should be aware of these differences. They can cause unexpected behaviour in your scripts. The following table demonstrates how the sort order for text varies. The “Orig” column shows a random assortment of characters. The following columns show the different sort orders generated by the PHP sort() function on both z/OS UNIX and Linux platforms:

Orig	**z/OS**	**Linux**
=		
A	:	0
0	=	9
Z	a	:
:	z	=
	A	A
z	Z	Z
9	0	z
Z	9	z

Other PHP functions affected by sort order differences between EBCDIC and ASCII are:

- strcmp()
- strncmp()
- strnatcmp()

9.4 Characters versus Code Points

One of the most common problems when porting scripts from other platforms to z/OS is caused by the technique of referring to characters using their code point (ordinal) value rather than their symbolic value. For example, consider the following code:

```
$x=ord(’A’);
if ($x == 0x41) {
       print "x is 'A' \n";
} else {
       print "x is not 'A' \n";
}
```

The `if` statement would only be true on an ASCII platform where the ordinal value of the character `A` is 0x41. If this code were run on z/OS UNIX, it would print `x is not 'A'` because the ordinal value of `A` on an EBCDIC code page is 0xC1.

Other PHP functions affected are:

- chr()
- count_chars()
- bin2hex()
- md5(), sha1(), crc32() hashing functions
- printf(), vsprintf(), vprintf(), fprintf(), sprintf() functions when printing using the %c format modifier

9.5 Newline ("\n")

The ordinal value of the newline ("\n") character is different between EBCDIC and ASCII. So, the symbolic representation of this character should always be used rather than the ordinal values. The following are incorrect and correct examples:

```
<?php
  print "Hello World \012"; # incorrect
  print "Hello World \n";   # correct
?>
```

The following table shows the ordinal value of the newline character in both EBCDIC and ASCII:

	Dec	Octal	hex
EBCDIC	21	025	0x15
ASCII	10	012	0xA

9.6 Non-contiguous Character Ranges

The EBCDIC code set does not have the alphabet arranged in a contiguous manner - there are gaps. For example, in ASCII, “A” to “Z” occupy 26 contiguous code points from 0x41 to 0x5A however in EBCDIC there are gaps in the sequence as follows:

- A-I occupy 0xC1 to 0xC9
- J-R occupy 0xD1 to 0xD9
- S-Z occupy 0xE2 to 0xE9

Similar gaps occur for the lower case characters too and the ordering of special characters is also different. Consequently, any PHP functions which accept a range will produce different results on z/OS UNIX, an EBCDIC-based platform, to other platforms which are ASCII based.

PHP Functions affected:

- trim(), ltrim(), and rtrim() when supplied a range (for example, trim($string, "a..z") will trim more than just the lower case characters “a” to “z” when run on EBCDIC)
- addcslashes() when supplied a range
- array range() function

9.7 Hash Function

PHP supports many hash functions. Some examples include MD5, SHA1, BASE64, CRC32, among others. For more information on these hash functions, consult the following RFCs:

Function name	**Corresponding RFC file**
MD5	RFC 1321 – The MD5 Message-Digest Algorithm

SHA1	RFC 3174 – US Secure Hash Algorithm (SHA1)
CRC32	
BASE64	RFC 2045 – Multipurpose Internet Mail Extensions (MIME) Part One:format of Internet Message Bodies

As previously mentioned the binary value of a string on z/OS UNIX is not equal to what it would be on a generic ASCII platform so the output from these hashing functions will differ. Consider the following simple example:

```
<?php
$str = 'apple';
var_dump(sha1($str1));
?>
```

On a Linux system the result is:

```
string(40) "d0be2dc421be4fcd0172e5afceea3970e2f3d940"
```

On a z/OS UNIX system the result is:

```
string(40) "cfdc0cb96f77e4272807cf2c1908b2da6b576898"
```

9.8 JSON Functions

On ASCII-based platforms such as Linux, the JSON functions json_encode() and json_decode() only work with UTF-8 (an ASCII superset code page) data. On z/OS UNIX however, these functions have been modified to work with both EBCDIC and UTF-8 data by adding a new optional flag to each command as follows:

```
string json_encode ( mixed $value [, string $input_encoding])
```

By default any input is assumed to be EBCDIC-encoded, i.e. IBM-1047 or IBM-037 code page. However, if the optional string argument `$input_encoding` is set to `"UTF8"`, then all input is assumed to be UTF-8 encoded as on ASCII based platforms. All output is in EBCDIC.

```
mixed json_decode ( string $json [, bool $assoc, string $output_encoding ] )
```

All input is assumed to be EBCDIC encoded and by default all output is also EBCDIC. However, if the optional string argument `$output_encoding` is set to `"UTF8"`, then all output will be UTF-8 encoded.

Note that `"UTF8"` is currently the only value supported by the optional arguments `$input_encoding` and `$output_encoding`.

9.9 XML Parser Functions

On z/OS UNIX, the XML parser functions will only correctly process input in EBCDIC. It will not understand iso-8859-1 or utf-8 documents, and the encoding tag in an XML document becomes meaningless.

Also, the functions utf8_encode() and utf8_decode() are not supported.

9.10 POSIX functions

Due to differences in the implementation of certain C library functions on z/OS UNIX, the output of some POSIX functions will be different to that on other platforms such as Linux:

- `gr_passwd` is not defined in the `grp.h` structure returned by the `getgrnam_r()` and `getgrgid_r()` system functions, so the arrays returned by the PHP functions `posix_getgrnam()` and `posix_getgrgid()` do not contain "`passwd`" entries.
- `pw_passwd` and `pw_gecos` are not defined in the `pwd.h` structure retuned by the `getpwnam_r()` system function, so the arrays returned by PHP functions `posix_getpwnam()` and `posix_getpwuid()` do not contain "`passwd`" or "`gecos`" entries.

9.11 Network Functions

The following pre-defined PHP constants are not available on z/OS UNIX:

- LOG_AUTHPRIV
- LOG_SYSLOG
- LOG_PERROR

9.12 File System Functions and Streams

The file system functions work with streams not just files. Streams can come from file systems or the network such as http servers, which are likely to provide data in UTF-8 or some other ASCII superset code page. Some of the file system functions parse the stream data. For example fgets() will look for EOL characters and only return information up to that EOL. fgetcsv() looks for EOL and also recognises white-space and meta-characters such as quotes and an escape character. There are 2 issues here:

- Stream data could not be EBCDIC, in which case it cannot be echoed directly or passed to extension functions that expect the data to be in runtime encoding. The script writer must perform a conversion on the data to ensure it is suitable. An example of how to convert is shown here:

```
<?php
// Use ini_get("unicode.runtime_encoding") to easily determine the runtime
encoding that is needed.
echo mb_convert_encoding(file_get_contents("http://www.example.com"),
ini_get("unicode.runtime_encoding"), "UTF-8");
?>
```

- This also affects the php://input stream, which is used for reading raw PUT and POST data from the HTTP request. Convert the data to the runtime encoding as shown below, or simply use **`zget('/request/input/transcoded')`**.

```
<?php
// Use $_SERVER['REQUEST_BODY_CHARSET'] to determine the charset of the
request body, as specified by the client. Defaults to "UTF-8".
echo mb_convert_encoding(file_get_contents("php://input"),
ini_get("unicode.runtime_encoding"), $_SERVER['REQUEST_BODY_CHARSET']);
?>
```

- In our current implementation, file system functions that perform a parsing operation will only work on the data in EBCDIC format (These APIs access the stream directly so there is no way to intercept and alter the stream data currently). The following functions are affected:
  - fgets()
  - fgetcsv()
  - file()
  - parse_ini_file()
- Some file system functions will also write runtime_encoding bytes out on your behalf which will make the creation of ASCII files problematic. This affects:
  - fputcsv()

9.13 System Functions

PHP calls system functions to implement some basic functions, such as mkdir(), fopen(), popen() and so on. Because z/OS UNIX is a UNIX standard compatible system, you should not encounter problems when using these functions except for EBCDIC to ASCII situations. It is possible, however, that you may encounter some difficulties if you attempt to implement some programming tricks. The following is an example:

```
#include <unistd.h>
int main(int argc, char *argv[]) {
    int c;
    extern char *optarg;
    while((c=getopt(argc,argv,"d:m:j:vht"))!=-1) {
        printf("%c-%s\n",c,optarg);
    }
}
```

Compile this program in Linux and z/OS UNIX and ensure that `Tgetopt` is the name of executable file:

Linux:
```
./Tgetopt -d 1 -m 2 -h 3
d-1
m-2
h-(null)
```

z/OS UNIX:
```
./Tgetopt -d 1 -m 2 -h 3
d-1
m-2
h-2
```

This inconsistency is due to differences in the implementation of the system function getopt(); you may get the same result when you try to use the PHP function getopt().

10. Debugging PHP Scripts

10.1 Logging Script Activity to File

Logging data or messages to a file is a simple way to track script behaviour without requiring a debug client and without contaminating the output of the script.

The function **`logToFile()`** below illustrates how to dump the contents of a variable to file, along with a timestamp and the file and line number on which **`logToFile()`** was called.

Note that the PHP function **`var_dump()`** usually writes its output straight to the response body. To catch the output in a variable, so we can write it to a file, we use PHP's output buffering functionality.

logToFile.php
```
<?php
date_default_timezone_set('UTC');

function logToFile($var) {
	// Dump $var into a string
	ob_start();
	var_dump($var);
	$data = ob_get_contents();
	ob_end_clean();

	// Gather calling filename and line number
	$backtrace = debug_backtrace();
	$file = $backtrace[0]['file'];
	$line = $backtrace[0]['line'];

	// Format message with timestamp
	$timeStamp = date('Ymd@H:m:s');
	$message = "$timeStamp - $file($line): $data";

	// Append message to log file
	file_put_contents("log.txt", $message, FILE_APPEND);
}
?>
```

It could be used as follows:

test.php
```
<?php
include('logToFile.php');
logToFile('Hello World');
logToFile(123);
?>
```

After executing **test.php** above, a file `log.txt` will be in the JVM's working directory with content like:
```
20081103@18:11:33 - /u/robinf/scripts/test.php(3): string(11) "Hello World"
20081103@18:11:33 - /u/robinf/scripts/test.php(4): int(123)
```

10.2 Debugging with Eclipse™ PDT

It is possible to remotely attach the Eclipse PDT XDebug client to the PHP engine. This allows the user to stop at breakpoints, step, evaluate expressions, inspect stack frames and inspect variable values from a development machine while a script executes on the server.

To debug with Eclipse PDT:

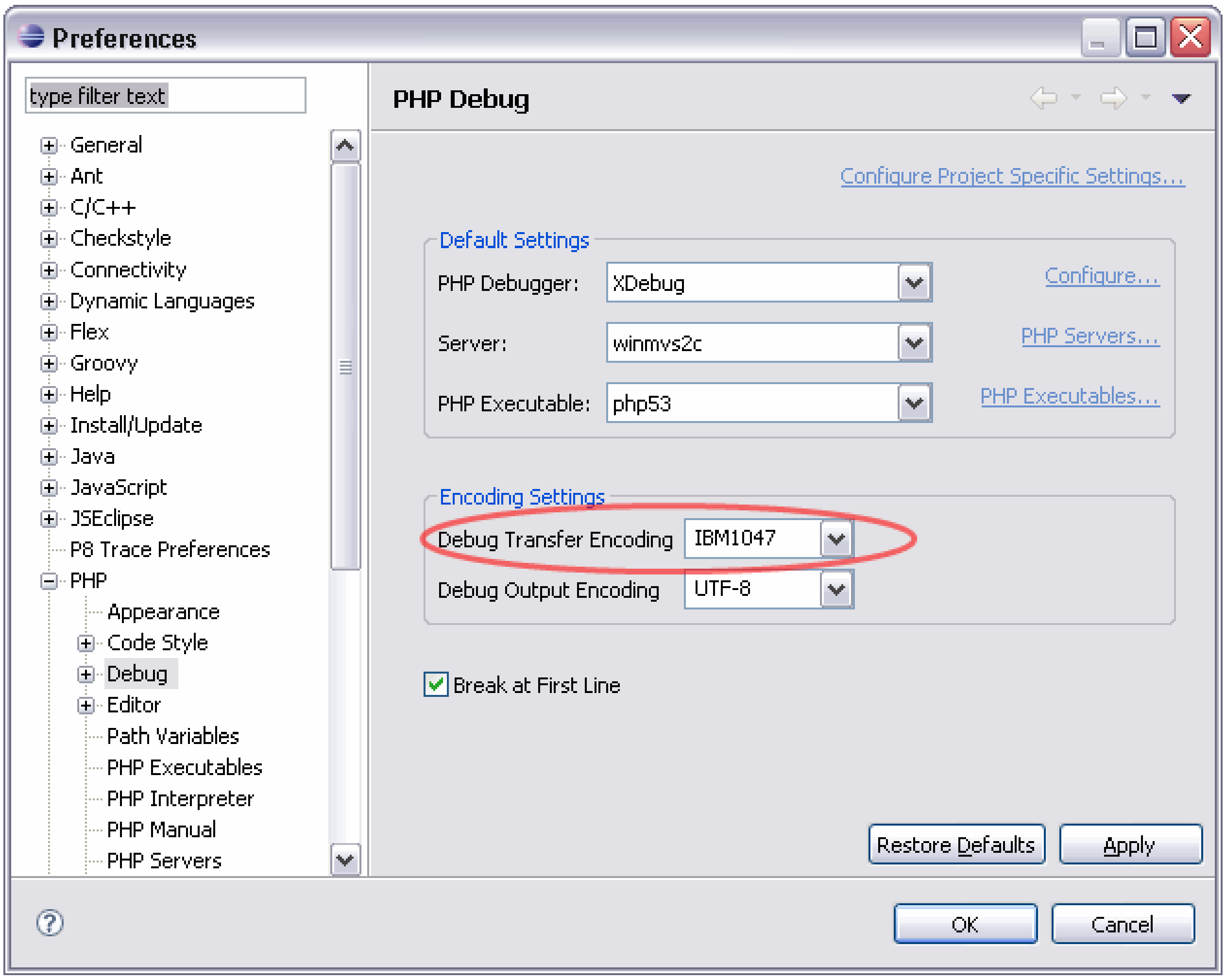
1. Download and install Eclipse PDT 2.0 or later: http://www.eclipse.org/pdt/downloads

2. Start eclipse PDT and load the scripts you wish to debug into the workspace. Place breakpoints as required.

3. Enable debug mode on the server by setting the JVM option below in your JVM profile, changing <*client_ip*> and <*client_port* > as appropriate (by default, the port used by PDT for XDebug is 9000, but this can be configured through Eclipse's preferences). Remember to phase out your JVMs so that the change to the JVM profile is picked up.

```
-Dp8.debug=idekey=ECLIPSE_DBGP&remotePort=<client_port>&remoteHost=<client_ip>
```

4. In the PHP debug preferences, set the "Debug Transfer Encoding" option to match match your unicode.runtime_encoding php.ini setting (which by default is is IBM-1047). This ensures string variable values will be displayed correctly by the debugger:



PHP Debug

Default Settings

PHP Debugger: XDebug

Server: winmvs2c

PHP Executable: php53

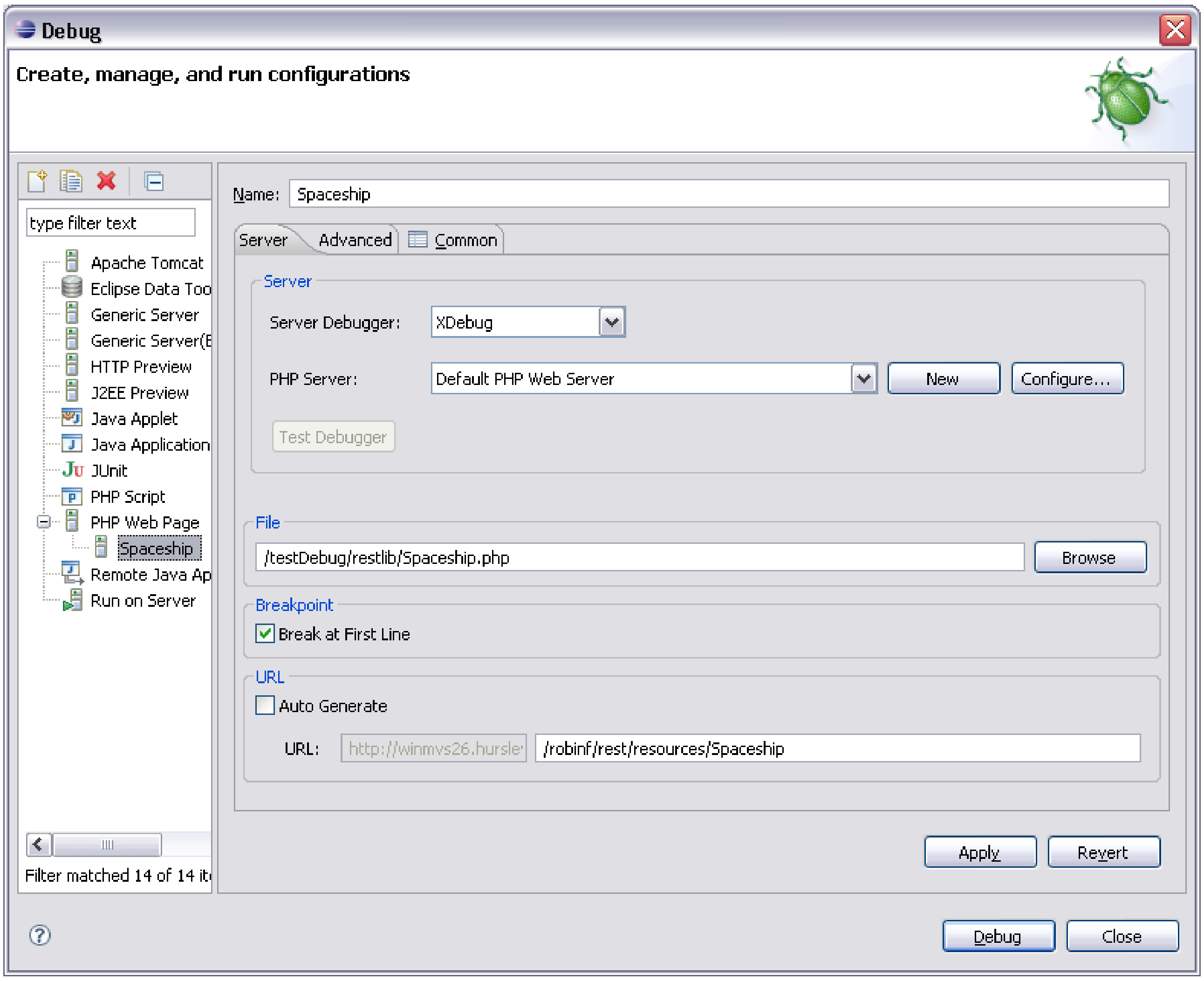
Encoding Settings

Debug Transfer Encoding IBM1047

Debug Output Encoding UTF-8

✓ Break at First Line

5. Create a PHP Web Page debug launcher with “Server Debugger” set to XDebug and the File and URL set as appropriate. For example:



Name: Spaceship

Server

Server Debugger: XDebug

PHP Server: Default PHP Web Server

File

/testDebug/restlib/Spaceship.php

Breakpoint

[✓] Break at First Line

URL

[ ] Auto Generate

URL: http://winmvs26.hursle /robinf/rest/resources/Spaceship

6. Click “debug”. This issues a request to the server by opening a browser at the specified URL. Eclipse PDT will then show the engine’s state in the PHP Debug perspective as breaks are encountered during execution on the server.

7. After the first request, the debug session will remain open. Requests will continue to communicate with the debug client until the session is explicitly ended (so even requests issued from clients other than the browser started by PDT can be debugged).

11. Troubleshooting

11.1 SupportPac User Forum

An online forum is provided so that users of CA1S can interact with each other and the CICS PHP development team:
http://www-128.ibm.com/developerworks/forums/forum.jspa?forumID=1537

Use it to better understand PHP in CICS, to debug problems, to make requests and to get answers. This forum is moderated by CICS PHP architects, developers, and testers.

When reporting unexpected behaviour, it may be useful to supply the contents of standard error and standard out, any error messages reported by the engine (including abend codes), as well as internal engine trace files for the requests that engendered the problem. The following sections explain how to collect this data.

11.2 Standard Out and Standard Error

The JVM's standard out and standard error streams will be redirected as defined in the JVM profile. For example, add the following to your JVM profile to write them to file under the JVM working directory, in directory `logs`:

```
STDOUT=logs/dfhjvmout -generate
STDERR=logs/dfhjvmerr -generate
```

The JVM working directory can also be configured in the JVM profile, by setting e.g.:
`WORK_DIR=/my/working/directory`

11.3 Abend Codes

The PHP handler will trigger a CICS abend when unexpected behaviour occurs. In the event of such a failure, an abend code may be reported in an HTTP 500 error message, and can be found in the JVM standard error log.

- **P8HA**: Error while reading the pipeline configuration from container DFH-HANDLERPLIST.
- **P8HB**: Error while encoding cookie data.
- **P8HC**: Task.getCurrentChannel() returned null, was expecting current Channel; unable to access pipeline configuration data or set HTTP response data.
- **P8HD**: Error retrieving HTTP method from the HTTP request object supplied by JCICS.
- **P8HE**: Error while attempting to notify client that the requested script was not found.
- **P8HF**: Fatal internal error while executing the script, or exception while starting or ending request.
- **P8HG**: Error while preparing response in containers DFHHTTPSTATUS, DFHMEDIATYPE and DFHRESPONSE.
- **P8HH**: Error retrieving data from the HTTP request object supplied by JCICS.
- **P8HI**: Error preparing HTTP request input data for consumption by PHP engine
- **P8HH**: Error retrieving request path from the HTTP request object supplied by JCICS.
- **P8HK**: Error while configuring XML parser before reading pipeline configuration.
- **P8HL**: Error while parsing pipeline configuration XML data.
- **P8HM**: Error while preparing PHP request based on pipeline configuration and HTTP request object supplied by JCICS.
- **P8HO**: Invalid request error while retrieving content from HTTP request object supplied by JCICS.
- **P8HP**: Request body not found while retrieving content from HTTP request object supplied by JCICS.
- **P8HQ**: Request body not HTTP data while retrieving content from HTTP request object supplied by JCICS.

- **P8HR**: Invalid request error while retrieving body character set from HTTP request object supplied by JCICS.
- **P8HS**: Bad request body encoding while decoding request input data.
- **P8HT**: Invalid request error while retrieving request content type to decode PUT data.
- **P8HU**: Invalid request error while attempting to roll back.
- **P8HV**: Invalid request error while attempting to commit
- **P8HZ**: Uncaught exception in the PHP handler.

11.4 Collecting PHP Engine Internal Trace Data

Internal tracing produces a set of binary files that describe the behaviour of the PHP engine over the course of one or more requests. When reporting a failure or unexpected behaviour, it is useful to include these files along with the JVM standard error and standard out logs.

While trace data is automatically dumped to file in the event of an internal engine failure, tracing can also be enabled manually. This is achieved via the properties file ***p8logging.properties***, which must be ASCII encoded and reside on the Java CLASSPATH. The following table defines the properties that can be set in this file. Keys are case sensitive. If no p8logging.properties file is found on the class path, then all the defaults apply.

Key	Description	Default
traceFilePrefix	Prefix to use for trace file names	p8Trace
traceCycleCount	Total number of trace files which are cycled. File .0 is the newest	100
traceRecordLimit	Number of trace records per file	10000
traceToFile	Send trace directly to file, bypassing the memory buffer	false
tracePushLevel	Trace level that automatically triggers the memory buffer to write out to a trace file	PHP_ERROR
javaDump	Whether a Java dump should be taken when an uncaught throwable trigger occurs	true
systemDumpLevel	trace level that triggers a J9 system dump	SEVERE
memoryRecordLimit	Number of trace records to be held in the cyclic memory buffer	1000
traceLevel	Level of trace records for all components to hold in memory or send to trace file. Valid values include DEBUG, INFO, WARNING and SEVERE.	INFO

For example, to consistently write the highest level of trace detail to file, set `traceLevel` to DEBUG and enable `traceToFile` by creating a p8logging.properties file on the CLASSPATH containing:

```
traceLevel = DEBUG
traceToFile = true
```

The above settings will significantly impact performance, so it is recommended to enable them only when gathering diagnostic data.

Note that changes to p8logging.properties will only be picked up after a JVM restart.

12. Known Issues and Limitations

- **Symptom**: Scripts that generate no content return responses with HTTP status code 200 OK, even though a different status is set in the script.
  Explanation: When the PHP handler attempts to send a response with an empty body to the client, any response HTTP status code set in the script is erroneously ignored. Instead, the response is always sent with HTTP 200 OK.

Resolution: This occurs only when the script generates no content, so it may be acceptable to temporarily work around the problem by adding white space to the output. This issue will be resolved in CICS TS 3.2 in APAR PK76485.

- **Symptom**: Attempting to return binary data from a script to the client in the response body results in the client receiving corrupt data.
  Explanation: All output from PHP scripts is treated as text data by the CICS pipeline process, regardless of the media type set in the Content-Type response header. Therefore, all script output is transcoded from the LOCALCCSID to UTF-8 before being sent to the client – including binary data.
  Resolution: As a temporary workaround, it may be possible to obtain the desired results by transcoding the data before sending it, with the expectation that CICS will transcode it again. This issue will be resolved in a future version of this SupportPac; the fix may require APAR PK76485.

- **Symptom**: Stale data is seen when calling getter methods on a commarea data structure.
  Explanation: The JZOS generated classes used to represent commarea data structures must be generated with the correct options.
  Resolution: Re-generate and recompile the Java class representing your data structure, ensuring that the following option is used: `genCache=false`

- **Symptom**: The commarea passed to the target CICS program contains low-values for fields which have not been set by the application, even though VALUE clauses are present in the COBOL source for those fields.
  Explanation: The JZOS generated classes do not contain initializers for the fields in a commarea if it is derived from a level 01 structure in the LINKAGE section of the source COBOL program.
  Resolution: Modify the COBOL program so that the required record structure is defined in the WORKING-STORAGE section of the source. Re-compile the program to create a new ADATA file. Re-generate and recompile the Java class representing your data structure.
  N.B. In this situation, it will probably be preferable to create a dummy COBOL program skeleton and insert or COPY the definition for the record, ensuring that the level 01 data definition for the record is in the WORKING-STORAGE section of the dummy program.

- **Symptom**: Real storage in subpool 2 key 8 progressively grows as HTTP requests are processed.
  Explanation: There is a known, minor memory leak in JCICS HttpRequest, which is used by this SupportPac.
  Resolution: The memory leak will be fixed in CICS TS 3.2 in APAR PK77018.

13. PHP Language Support Reference

Support for the PHP language is provided by a Java-based runtime package. The runtime supports almost all the language features of PHP 5.2 and a subset of the functions provided by PHP extensions.

13.1 Core Functions

The tables below list the extension functions supported in the runtime for PHP included in CA1S. The complete list of available functions and classes can be accessed programmatically with functions get_defined_functions() and get_declared_classes().

Array Functions		
array	array_pop	current
array_change_key_case	array_product	each
array_chunk	array_push	end
array_combine	array_rand	extract
array_count_values	array_reduce	in_array

array_diff	array_reverse	key
array_diff_assoc	array_search	key_exists
array_diff_key	array_shift	krsort
array_diff_uassoc	array_slice	ksort
array_diff_ukey	array_splice	list
array_fill	array_sum	natcasesort
array_fill_keys	array_udiff	natsort
array_filter	array_udiff_assoc	next
array_flip	array_udiff_uassoc	pos
array_intersect	array_uintersect	prev
array_intersect_assoc	array_uintersect_assoc	range
array_intersect_key	array_uintersect_uassoc	reset
array_intersect_uassoc	array_unique	rsort
array_intersect_ukey	array_unshift	shuffle
array_key_exists	array_values	sizeof
array_keys	array_walk	sort
array_map	array_walk_recursive	uasort
array_merge	arsort	uksort
array_merge_recursive	asort	usort
array_multisort	compact	
array_pad	count	

CICS Specific Functions (these functions are specific to this SupportPac – see previous sections for more information)		
CICSProgram::link	CICSTask::commit	zget
CICSProgram::link_sync	CICSTask::rollback	zlist

Class & Object Functions		
call_user_method	get_class_vars	interface_exists
call_user_method_array	get_declared_classes	is_a
class_exists	get_declared_interfaces	is_subclass_of
get_class	get_object_vars	method_exists
get_class_methods	get_parent_class	property_exists

Date & Time Functions		
checkdate	date_sunset	localtime
date	getdate	microtime
date_create	gettimeofday	mktime
date_default_timezone_get	gmdate	strftime
date_default_timezone_set	gmmktime	strtotime

date_sun_info	gmstrftime	time
date_sunrise	idate	

Error Handling and Logging Functions		
debug_backtrace	error_reporting	set_exception_handler
debug_print_backtrace	restore_error_handler	trigger_error
error_get_last	restore_exception_handler	user_error
error_log	set_error_handler	

Filesystem Functions		
basename	filemtime	is_writeable
chgrp	fileowner	link
chmod	fileperms	linkinfo
chown	filesize	lstat
clearstatcache	filetype	mkdir
copy	flock	move_uploaded_file
dirname	fopen	parse_ini_file
disk_free_space	fpassthru	pathinfo
diskfreespace	fputcsv	pclose
fclose	fputs	popen
feof	fread	readfile
fflush	fseek	readlink
fgetc	ftell	realpath
fgetcsv	ftruncate	rename
fgets	fwrite	rewind
file	glob	rmdir
file_exists	is_dir	stat
file_get_contents	is_executable	symlink
file_put_contents	is_file	tempnam
fileatime	is_link	touch
filectime	is_readable	umask
filegroup	is_uploaded_file	unlink
fileinode	is_writable	

Function Handling Functions		
call_user_func	func_get_args	register_shutdown_function
call_user_func_array	func_num_args	register_tick_function
create_function	function_exists	unregister_tick_function
func_get_arg	get_defined_functions	

Mathematical Functions		
abs	deg2rad	min
acos	exp	mt_getrandmax
acosh	expm1	mt_rand
asin	floor	mt_srand
asinh	fmod	octdec
atan	getrandmax	pi
atan2	hexdec	pow
atanh	hypot	rad2deg
base_convert	is_finite	rand
bindec	is_infinite	round
ceil	is_nan	sin
cos	lcg_value	sinh
cosh	log	sqrt
decbin	log10	srand
dechex	log1p	tan
decoct	max	tanh

Multi-byte String Functions		
mb_convert_encoding	mb_split	mb_strrpos
mb_decode_mimeheader	mb_stripos	mb_strstr
mb_encode_mimeheader	mb_stristr	mb_strtolower
mb_ereg	mb_strlen	mb_strtoupper
mb_ereg_replace	mb_strpos	mb_substitute_character
mb_internal_encoding	mb_strrchr	mb_substr
mb_regex_encoding	mb_strrichr	mb_substr_count
mb_regex_set_options	mb_strripos	

Network Functions		
closelog	header	setcookie
define_syslog_variables	headers_list	setrawcookie
fsockopen	headers_sent	socket_get_status
gethostbyaddr	ip2long	socket_set_blocking
gethostbyname	long2ip	socket_set_timeout
gethostbynamel	pfsockopen	syslog

Output Control Functions		
flush	ob_get_clean	ob_get_status
ob_clean	ob_get_contents	ob_implicit_flush

ob_end_clean	ob_get_flush	ob_list_handlers
ob_end_flush	ob_get_length	ob_start
ob_flush	ob_get_level	

JSON Functions		
json_decode	json_encode	

PDO Functions (See also the PDO section of this document)		
pdo::construct	pdo::prepare	pdostatement::closecursor
pdo::exec	pdo::query	pdostatement::execute
pdo::getattribute	pdo::quote	pdostatement::fetch
pdo::getavailabledrivers	pdo::setattribute	pdostatement::fetchall
pdo::lastinsertid	pdostatement::bindvalue	pdostatement::rowcount

Reflection Classes (not all methods are available on all classes)		
ReflectionClass	ReflectionProperty	ReflectionParameter
ReflectionObject	ReflectionMethod	ReflectionFunction

SPL Classes		
ArrayAccess	BadFunctionCallException	OutOfRangeException
ArrayIterator	BadMethodCallException	OverflowException
ArrayObject	InvalidArgumentException	RangeException
Countable	LengthException	RuntimeException
Iterator	LengthException	UnderflowException
Serializable	LogicException	UnexpectedValueException
Traversable	OutOfBoundsException	

String Functions		
addcslashes	nl_langinfo	stripos
addslashes	number_format	stripslashes
bin2hex	ord	stristr
chop	parse_str	strlen
chr	print	strnatcasecmp
chunk_split	printf	strnatcmp
convert_cyr_string	quoted_printable_decode	strncasecmp
convert_uudecode	quotemeta	strncmp
convert_uuencode	rtrim	strpbrk
count_chars	setlocale	strpos
crc32	sha1	strrchr

echo	sha1_file	strrev
explode	similar_text	strripos
fprintf	soundex	strrpos
get_html_translation_table	sprintf	strspn
hebrev	sscanf	strstr
hebrevc	str_ireplace	strtok
html_entity_decode	str_pad	strtolower
htmlentities	str_repeat	strtoupper
htmlspecialchars	str_replace	strtr
htmlspecialchars_decode	str_rot13	substr
implode	str_shuffle	substr_compare
join	str_split	substr_count
levenshtein	str_word_count	substr_replace
localeconv	strcasecmp	trim
ltrim	strchr	ucfirst
md5	strcmp	ucwords
md5_file	strcoll	vfprintf
metaphone	strcspn	vprintf
money_format	strip_tags	vsprintf
nl2br	stripcslashes	wordwrap

Variable Handling Functions		
debug_zval_dump	is_double	is_string
doubleval	is_float	isset
empty	is_int	print_r
floatval	is_integer	serialize
get_defined_vars	is_long	settype
get_resource_type	is_null	strval
gettype	is_numeric	unserialize
intval	is_object	unset
is_array	is_real	var_dump
is_bool	is_resource	var_export
is_callable	is_scalar	

XML Parser Functions	
xml_error_string	xml_parser_set_option
xml_get_current_byte_index	xml_set_character_data_handler
xml_get_current_column_number	xml_set_default_handler
xml_get_current_line_number	xml_set_element_handler
xml_get_error_code	xml_set_end_namespace_decl_handler

xml_parse	xml_set_external_entity_ref_handler
xml_parse_into_struct	xml_set_notation_decl_handler
xml_parser_create	xml_set_object
xml_parser_create_ns	xml_set_processing_instruction_handler
xml_parser_free	xml_set_start_namespace_decl_handler
xml_parser_get_option	xml_set_unparsed_entity_decl_handler

13.2 php.ini Directives

A number of configuration directives are supported in the php.ini file which can be set to alter behaviour when executing PHP scripts.

If a directive is in the list but is not supported, then you cannot change that directive in the php.ini file, and only the default behaviour is provided.

Directive	Explanation	Supported	Default Value	Differences
zend.ze1_compatibility_mode	Enable compatibility mode with older versions of PHP (PHP 4.x)	No	Off	
short_open_tag	Allow the <? tag. Otherwise, only <?php and <script> tags are recognized. NOTE: Using short tags should be avoided when developing applications or libraries that are meant for redistribution, or deployment on PHP servers which are not under your control, because short tags may not be supported on the target server. For portable, redistributable code, be sure not to use short tags.	Yes	Off	
asp_tags	Allow ASP-style <% %> tags.	Yes	Off	
	The number of significant digits displayed in floating point numbers.	Yes	14	
y2k_compliance	Enforce year 2000 compliance (will cause problems with non-compliant browsers)	No	On	
output_buffering	Output buffering allows you to send header lines (including cookies) even after you send body content, at the price of slowing PHP's output layer a bit. You can enable output buffering during runtime by calling the output buffering functions. You can also enable output buffering for all files by setting this directive to On. If you wish to limit the size of the buffer to a certain size - you can use a maximum number of bytes instead of 'On', as a value for this directive (e.g., output_buffering=4096).	No	Off	
output_handler	You can redirect all of the output of your scripts to a function. Setting any output handler automatically turns on output buffering. Note: People who wrote portable scripts should not depend on this ini directive. Instead, explicitly set the output handler using ob_start(). Using this ini directive may cause problems unless you know what script is doing.	No	Off	
implicit_flush	Implicit flush tells PHP to tell the output layer to	No	Off	

Directive	Explanation	Supported	Default Value	Differences
	flush itself automatically after every output block. This is equivalent to calling the PHP function flush() after each and every call to print() or echo() and each and every HTML block. Turning this option on has serious performance implications and is generally recommended for debugging purposes only.			
unserialize_callback_func	The unserialize callback function will be called (with the undefined class' s name as parameter), if the unserializer finds an undefined class which should be instantiated. A warning appears if the specified function is not defined, or if the function doesn't include/implement the missing class. So only set this entry if you really want to implement such a callback-function.	Yes		
serialize_precision	When floats & doubles are serialized, store serialize_precision significant digits after the floating point. The default value ensures that when floats are decoded with unserialize, the data will remain the same.	Yes	Off	
allow_call_time_pass_reference	Whether to enable the ability to force arguments to be passed by reference at function call time. This method is deprecated and is likely to be unsupported in future versions of PHP. The encouraged method of specifying which arguments should be passed by reference is in the function declaration. You're encouraged to try and turn this option Off and make sure your scripts work properly with it in order to ensure they will work with future versions of the language (you will receive a warning each time you use this feature, and the argument will be passed by value instead of by reference).	No	Off	Call time pass by reference is not supported.
safe_mode	Safe Mode	No	Off	
safe_mode_gid	By default, Safe Mode does a UID compare check when opening files. If you want to relax this to a GID compare, then turn on safe_mode_gid.	No	Off	
safe_mode_include_dir	When safe_mode is on, UID/GID checks are bypassed when including files from this directory and its subdirectories. (directory must also be in include_path or full path must be used when including)	No		
safe_mode_exec_dir	When safe_mode is on, only executables located in the safe_mode_exec_dir will be allowed to be executed via the exec family of functions.	No		
safe_mode_allowed_env_vars	Setting certain environment variables may be a potential security breach. This directive contains a comma-delimited list of prefixes. In Safe Mode, the user may only alter environment variables whose names begin with the prefixes supplied here. By default, users will only be able to set environment variables that begin with PHP_ (e.g. PHP_FOO=BAR). Note: If this directive is empty, PHP will let the user modify ANY environment variable!	No		putenv() is not supported so no environment variables can be changed.
safe_mode_protec	This directive contains a comma-delimited list of	No		putenv() is not

Directive	Explanation	Supported	Default Value	Differences
ted_env_vars	environment variables that the end-user won't be able to change using putenv(). These variables will be protected even if safe_mode_allowed_env_vars is set to allow to change them.			supported so no environment variables can be changed
open_basedir	open_basedir, if set, limits all file operations to the defined directory and below. This directive makes most sense if used in a per-directory or per-virtualhost web server configuration file. This directive is *NOT* affected by whether Safe Mode is turned On or Off.	Yes		
disable_functions	This directive allows you to disable certain functions for security reasons. It receives a comma-delimited list of function names. This directive is *NOT* affected by whether Safe Mode is turned On or Off.	No		
disable_classes	This directive allows you to disable certain classes for security reasons. It receives a comma-delimited list of class names. This directive is *NOT* affected by whether Safe Mode is turned On or Off.	No		
highlight.string	Colors for Syntax Highlighting mode. Anything that's acceptable in <span style="color: ???????"> would work.	Yes	#DD0000	
highlight.comment	Colors for Syntax Highlighting mode. Anything that's acceptable in <span style="color: ???????"> would work.	Yes	#FF8800	
highlight.keyword	Colors for Syntax Highlighting mode. Anything that's acceptable in <span style="color: ???????"> would work.	Yes	#007700	
highlight.bg	Colors for Syntax Highlighting mode. Anything that's acceptable in <span style="color: ???????"> would work.	Yes	#FFFFFF	
highlight.default	Colors for Syntax Highlighting mode. Anything that's acceptable in <span style="color: ???????"> would work.	Yes	#0000BB	
highlight.html	Colors for Syntax Highlighting mode. Anything that's acceptable in <span style="color: ???????"> would work.	Yes	#000000	
ignore_user_abort	If enabled, the request will be allowed to complete even if the user aborts the request. Consider enabling it if executing long request, which may end up being interrupted by the user or a browser timing out.	No	On	
expose_php	Decides whether PHP may expose the fact that it is installed on the server (e.g. by adding its signature to the Web server header). It is no security threat in any way, but it makes it possible to determine whether you use PHP on your server or not.	No	Off	
max_execution_time	Maximum execution time of each script, in seconds	Yes	30	
max_input_nesting_level	Maximum nesting of arrays created from POST/GET data	Yes	64	
error_reporting	error_reporting is a bit-field. “Or” up each number	Yes		

Directive	Explanation	Supported	Default Value	Differences
	to get desired error reporting level. • E_ALL - All errors and warnings (doesn't include E_STRICT) • E_ERROR - fatal run-time errors E_RECOVERABLE_ERROR - almost fatal run-time errors • E_WARNING - run-time warnings (non-fatal errors) • E_PARSE - compile-time parse errors • E_NOTICE - run-time notices (these are warnings which often result from a bug in your code, but it's possible that it was intentional (e.g., using an uninitialized variable and relying on the fact it's automatically initialized to an empty string) • E_STRICT - run-time notices, enable to have PHP suggest changes to your code which will ensure the best interoperability and forward compatibility of your code • E_CORE_ERROR - fatal errors that occur during PHP's initial startup • E_CORE_WARNING - warnings (non-fatal errors) that occur during PHP's initial startup • E_COMPILE_ERROR - fatal compile-time errors • E_COMPILE_WARNING - compile-time warnings (non-fatal errors) • E_USER_ERROR - user-generated error message • E_USER_WARNING - user-generated warning message • E_USER_NOTICE - user-generated notice message Examples: Show all errors, except for notices and coding standards warnings: error_reporting = E_ALL & ~E_NOTICE Show all errors, except for notices: error_reporting = E_ALL & ~E_NOTICE | E_STRICT Show only errors: error_reporting = E_COMPILE_ERROR| E_RECOVERABLE_ERROR|E_ERROR| E_CORE_ERROR		E_ALL & ~E_NOTICE & ~E_STRICT	
display_errors	Print out errors (as a part of the output). For production web sites, you're strongly encouraged to turn this feature off, and use error logging instead (see below). Keeping display_errors enabled on a production web site may reveal security information to end users, such as file	Yes	On	Value of stderr is not supported.

Directive	Explanation	Supported	Default Value	Differences
	paths on your Web server, your database schema or other information. possible values for display_errors: Off - Do not display any errors stderr - Display errors to STDERR (affects only CGI/CLI binaries!)display_errors = "stderr" stdout (On) - Display errors to STDOUT			
display_startup_errors	Even when display_errors is on, errors that occur during PHP's startup sequence are not displayed. It's strongly recommended to keep display_startup_errors off, except for when debugging.	Yes	Off	
log_errors	Log errors into a log file (server-specific log, stderr, or error_log) As stated above, you're strongly advised to use error logging in place of error displaying on production web sites.	Yes	Off	
log_errors_max_len	Set maximum length of log_errors. In error_log information about the source is added. The default is 1024 and 0 allows to not apply any maximum length at all.	Yes	1024	
ignore_repeated_errors	Do not log repeated messages. Repeated errors must occur in same file on same line until ignore_repeated_source is set true.	Yes	Off	
ignore_repeated_source	Ignore source of message when ignoring repeated messages. When this setting is On you will not log errors with repeated messages from different files or source lines.	Yes	Off	
track_errors	Store the last error/warning message in $php_errormsg (boolean).	Yes	Off	
html_errors	Disable the inclusion of HTML tags in error messages. Note: Never use this feature for production boxes.	Yes	On	
docref_root	If html_errors is set On PHP produces clickable error messages that direct to a page describing the error or function causing the error in detail. You can download a copy of the PHP manual from http://www.php.net/docs.php and change docref_root to the base URL of your local copy including the leading '/'. You must also specify the file extension being used including the dot. Note: Never use this feature for production boxes.	Yes		
docref_ext	If html_errors is set On PHP produces clickable error messages that direct to a page describing the error or function causing the error in detail. You must specify the file extension being used including the dot.	Yes		
error_prepend_string	String to output before an error message.	Yes		
error_append_string	String to output after an error message.	Yes		
error_log	Log errors to specified file.	Yes		A value of syslog is not supported.
arg_separator.output	The separator used in PHP generated URLs to separate arguments.	Yes	&	
arg_separator.inpu	List of separator(s) used by PHP to parse input	Yes	&	

Directive	Explanation	Supported	Default Value	Differences
t	URLs into variables. NOTE: Every character in this directive is considered as separator!			
register_globals	Whether or not to register the EGPCS variables as global variables. You may want to turn this off if you don't want to clutter your scripts' global scope with user data. This makes most sense when coupled with track_vars - in which case you can access all of the GPC variables through the $HTTP_*_VARS[], variables. You should do your best to write your scripts so that they do not require register_globals to be on; Using form variables as globals can easily lead to possible security problems, if the code is not very well thought of.	No	Off	
register_long_arrays	Whether or not to register the old-style input arrays, HTTP_GET_VARS and friends. If you're not using them, it's recommended to turn them off, for performance reasons.	No	Off	
register_argc_argv	This directive tells PHP whether to declare the argv&argc variables (that would contain the GET information). If you don't use these variables, you should turn it off for increased performance.	No	Off	
auto_globals_jit	When enabled, the SERVER and ENV variables are created when they're first used (Just In Time) instead of when the script starts. If these variables are not used within a script, having this directive on will result in a performance gain. The PHP directives register_globals, register_long_arrays, and register_argc_argv must be disabled for this directive to have any affect.	No	On	
include_path	Path to search for include files LINIX: "/path1:/path2" Windows: "\path1;\path2"	Yes		
user_dir	The directory under which PHP opens the script using /~username used only if nonempty.	No		
extension_dir	Directory in which the native loadable extensions (modules) reside. The JVM variable java.library.path is also searched for native loadable extensions. Java loadable extensions are found using the classpath and not this directive.	Yes		Only used to find native loadable extensions (dll, so, dylib) and not extensions written in Java. Native loadable extensions can also be in a directory specified on the java.library.path.
enable_dl	Whether or not to enable the dl() function.	Yes	On	
file_uploads	Whether to allow HTTP file uploads.	No	On	
upload_tmp_dir	Temporary directory for HTTP uploaded files (will use system default if not specified).	No		
upload_max_filesize	Maximum allowed size for uploaded files.	Yes	2M	
extension	Specifies extensions to be loaded on startup. For extensions written in Java, use:	Yes		

Directive	Explanation	Supported	Default Value	Differences

	`extension=package.java_class_name`
	For extensions written in C, use:
	`extension=extension_name`
	For example: `extension=php_gd`
	
	The appropriate filesystem extension (.dll, .so) will automatically be appended. Note that it should be the name of the module only; no directory information needs to go here. Specify the location of the extension with the extension_dir directive above.			

unicode.script_encoding	Script encoding defines how the PHP file is encoded. This must be correctly set in order for string literals and PHP names, such as function, variable and class names, to be correctly parsed by the runtime.	Yes	UTF-8	
unicode.runtime_encoding	Runtime encoding is used when a PHP string is converted into a Java string. Internally, the runtime preserves the ability to store binary data in a PHP string, which is essential for full support of the PHP 5 language.	Yes	UTF-8	Must match LOCALCCSID, with “IBM-” prefix.
date.timezone	Defines the default timezone used by the date functions	Yes		
date.default_longitude		Yes		
date.default_latitude		Yes	31.77	
date.sunset_zenith		Yes		
date.sunrise_zenith		Yes	90.58	
iconv.output_encoding	Default output encoding for iconv.	No		iconv not available on CICS.
iconv.input_encoding	Default input encoding for iconv.	No		iconv not available on CICS.
iconv.internal_encoding	Default internal encoding for iconv.	no		iconv not available on CICS.
pcre.backtrack_limit	PCRE library backtracking limit.	Yes		
pcre.recursion_limit	PCRE library recursion limit. Please note that if you set this value to a high number you may consume all the available process stack and eventually crash PHP (due to reaching the stack size limit imposed by the Operating System).	Yes		
mysql.allow_persistent	Allow or prevent persistent links.	No		The MySQL extension is not supported on CICS.
mysql.max_persistent	Maximum number of persistent links. -1 means no limit.	No		
mysql.max_links	Maximum number of links (persistent + non-persistent). -1 means no limit.	No		
mysql.default_port	Default port number for mysql_connect(). If unset, mysql_connect() will use the $MYSQL_TCP_PORT or the mysql-tcp entry in /etc/services or the compile-time value defined	No		

Directive	Explanation	Supported	Default Value	Differences
	MYSQL_PORT (in that order). Win32 will only look at MYSQL_PORT.			
mysql.default_socket	Default socket name for local MySQL connects. If empty, uses the built-in MySQL defaults.	No		
mysql.default_host	Default host for mysql_connect() (doesn't apply in safe mode).	No		
mysql.default_user	Default user for mysql_connect() (doesn't apply in safe mode).	No		
mysql.default_password	Default password for mysql_connect() (doesn't apply in safe mode). Note that this is generally a *bad* idea to store passwords in this file. *Any* user with PHP access can run 'echo get_cfg_var("mysql.default_password") and reveal this password! And of course, any users with read access to this file will be able to reveal the password as well.	No		
mysql.connect_timeout	Maximum time (in seconds) for connect timeout. -1 means no limit	No		
mysql.trace_mode	Trace mode. When trace_mode is active (=On), warnings for table/index scans and SQL-Errors will be displayed.	No		
session.*	Sessions are not supported on CICS.	No		
mbstring.internal_encoding	internal/script encoding. Some encoding cannot work as internal encoding. (e.g. SJIS, BIG5, ISO-2022-*)	Yes		
gd.jpeg_ignore_warning	Tell the jpeg decode to libjpeg warnings and try to create a gd image. The warning will then be displayed as notices disabled by default.	No		GD not available on CICS
optimization_level	This sets the level of optimization performed on the PHP code when it is compiled for execution. The default value provides the best performance while, at the same time, being safe for all applications. It may be set to "int" to force interpreted execution which performs more slowly.	Yes		
code_cache	Enables the use of an in memory cache of compiled PHP code. This eliminates the parsing and compilation step for the second and subsequent run of a PHP file. It provides a performance improvement for repeatedly executed files. Additional heap space is required to hold this cache. The space required is dependant on the number and complexity of files that it holds.	Yes	On	
code_cache_limit	Sets a limit on the number of files that will be held in the code cache. When this limit is reached entries are removed on a least recently used basis in order to accommodate new entries.	Yes	400	
persistent_code_cache	Enables the use of a filesystem store of compiled PHP code, this is an extension of the memory based code cache, however it is not limited in size and persists between JVM restarts.	Yes	Off	
persistent_code_cache_dir	Specifies the directory in which the persistent code cache is written.	Yes	.	

13.3 Superglobals

Magic Quotes

The magic_quotes_gpc setting is **not supported in the CICS environment**. Instead, GET/POST/COOKIE data should be explicitly sanitized in the application code.

Supported Superglobals

The following superglobals are available to PHP scripts running in the CICS environment:

- $GLOBALS
- $_ENV
- $_GET: See note on Magic Quotes above.
- $_POST: See note on Magic Quotes above.
- $_COOKIE: See note on Magic Quotes above.
- $_REQUEST
- $_SERVER: Contains information about the server and the request. The following elements are supported:
  - SERVER_ADDR
  - SERVER_NAME
  - SERVER_SOFTWARE
  - SERVER_PROTOCOL
  - SERVER_PORT
  - REQUEST_METHOD
  - REQUEST_TIME
  - REQUEST_BODY_CHARSET – this is specific to CICS. Contains the encoding of the raw PUT or POST data as defined in the request. If the encoding is not specified in the request, defaults to "ISO-8859-1" as per rfc2616 section 3.7.1.
  - SAPI_VERSION – this is specific to CICS. Contains the CA1S version string.
  - CONTENT_TYPE
  - CONTENT_LENGTH
  - HTTP_ACCEPT
  - HTTP_ACCEPT_CHARSET
  - HTTP_ACCEPT_ENCODING
  - HTTP_ACCEPT_LANGUAGE
  - HTTP_CONNECTION
  - HTTP_HOST
  - HTTP_USER_AGENT
  - HTTP_REFERER
  - HTTPS
  - REMOTE_ADDR
  - REMOTE_HOST
  - QUERY_STRING
  - DOCUMENT_ROOT
  - SCRIPT_FILENAME
  - SCRIPT_NAME
  - PHP_SELF
  - REQUEST_URI
  - argv
  - argc

Note that the *$_SESSION* superglobal and the related session functions are not supported.

13.4 General Differences in Behaviour

There are some differences between the behaviour of the PHP language support provided in this SupportPac and the behaviour of running a PHP script on the runtime available from http://php.net.

Description	Behaviour in php.net	Behaviour in CICS	Notes
Notice on array to string conversion	Behaviour is inconsistent (5.2.1).	Always outputs notice.	
Modifying array during foreach loop while using `&$value` syntax	Behaviour is inconsistent (5.2.1).	Changes to array always reflect upon `$value`.	
Expansion of complex variables inside strings	`$a[0]` expands to the value of `$a[0]`, but `$a[0][0]` expands to `Array[0]`.	Expands to the value of the variable in both cases.	
Parsing invalid UTF-8 strings `htmlentities(" Le CafÎ˜< CafÎ˜ <");`.	Ignores just the first `xE9` character (e acute in cp1252).	Both `xE9` character are ignored.	

`var_dump` of array containing a reference to a value, where there are no longer any other references to that value.	`array(1) {`
	`  [0]=>`
	`  &int(123456789)`
	`}`
	
	(note the ampersand)
		`array(1) {`
		`  [0]=>`
		`  int(123456789)`
		`}`
			This is a trivial example of a consequence of a garbage-collection model over reference counting. Reference counting here allows php.net to demote a reference to a value when its reference count falls to one, whereas in the runtime for PHP in this SupportPac, there is no way of knowing that there is only one reference to the value. This also affects backtrace dumps from `Exception` objects.

Difference in object handle in output of `var_dump`.	`object(stdClass)#1 (1) {`
	`        ["a"]=>`
	`        int(0)`
	`}`
		`object(stdClass)#2 (1) {`
		`        ["a"]=>`
		`        int(0)`
		`}`
			Each object has a handle/ID which is exposed by `var_dump($object)` after the `#` sign. In the runtime for PHP in this SupportPac, we cannot guarantee to use the same ID as php.net, where this ID is derived from the way the object is allocated in memory.

Using `__FUNCTION__` within an include within a function declaration.	`__FUNCTION__` evaluates to `""` (5.2.1).	`__FUNCTION__` evaluates to the name of the declaring function.	php.net's behaviour contradicts the manual entry for `include`: *If the include occurs*

Description	Behaviour in php.net	Behaviour in CICS	Notes
			inside a function within the calling file, then all of the code contained in the called file will behave as though it had been defined inside that function.
Superglobals, ampersands, `var_dump` and `$GLOBALS`.	Superglobals are not preceded by & in `$GLOBALS` (they *are not* shown as references) (5.2.1).	Superglobals are preceded by `&` in `$GLOBALS` (they are shown as references)	The runtime for PHP in this SupportPac handles superglobals by injecting references to them at all scopes, including global scope. Therefore, superglobals are shown as references in `$GLOBALS`.
Strict warnings when implicitly initialising objects.	`$obj->b=1;` emits a strict warning but `$obj->b[0]=1;` does not (5.2.3).	Both `$obj->b=1;` and `$obj->b[0]=1;` emit a strict warning.	`$obj->b=X` and `$obj->b[0]=X` both implicitly initialise `$obj` to an instance of `stdClass`, but in php.net only the first emits the warning: `Strict Standards: Creating default object from empty value`.
Reference counts off by one.	`$a=array(1);` Reference count of element will be one.	The reference count will be two.	This happens because a variable is created in the *data section* of the script and has an initial reference count of one. When this is assigned into an array, the reference count is incremented to two.
Size of integers.	32 bit integers on 32 bit platforms. 64 bit integers on 64 bit platforms.	32 bit integers on all platforms.	
The runtime for PHP in this SupportPac does not support short tags and <script> tags.			
Passing temporary value by reference to extension functions.	In php.net, attempting to pass a temporary value by reference to either a userspace or extension function causes a fatal error before execution of the script begins.	Consistent with this behaviour for userspace functions, but for extension functions the fatal error is not raised until the function call is reached.	
Parse error messages will include language construct not token	`Parse error: syntax error, unexpected T_FOR in <testcase path> on line 3`	`Parse error: syntax error, unexpected 'for' in <testcase path> on line 3`	

Description	Behaviour in php.net	Behaviour in CICS	Notes
names.			
`property_exists` behaviour.	php.net 5.2 tries to respect visibility of properties and has bugs. php.net 5.3 ignores scope and visibility.	Matches php.net 5.3 behaviour and matches logic for `method_exists` function.	

14. Version History

SupportPac CA1S v1.1

- Removed dependency on z/OS v1.9 to allow compatibility with z/OS v1.8.
- Performance enhancement to json_encode() and json_decode() which reduces IO operations during code-page conversions within those functions.
- Fix to ensure line endings are encoded correctly in responses which include a content-type header starting with "application/" and a charset parameter suffix.
- Several minor improvements and corrections to the documentation.

SupportPac CA1S v1.0

- Initial release

15. Legal Notices

15.1 Trademarks