

**Release 1862**

**Service Request 82205  
Time Reporting Web Service**

**Detail Design**

May 26, 2009  
Prepared by Maxine Gerber

Information Resources & Communications  
Office of the President  
University of California

# Table of Contents

<b>Introduction .....</b>	<b>1</b>
Service Request 82205.....	1
<b>Overview of CICS Web Services .....</b>	<b>1</b>
<b>Overview of PPS Web Services Security.....</b>	<b>1</b>
<b>Overview of PPS Modifications .....</b>	<b>1</b>
PPS CICS Infrastructure .....	1
Web Services Programs .....	1
Web Artifacts.....	1
Tokens .....	2
IDTC/EDHC Process.....	2
<b>Programs .....</b>	<b>2</b>
New and Modified Programs.....	2
UC0\$WBST (new) .....	2
PPWEBSEC (new) .....	2
PPXMLPRS (new).....	3
PPDELTS (new).....	3
UCROUTER.....	3
PPGETTIM.....	4
PPWEDHC .....	4
PPWIDTC.....	4
PS002 (new).....	4
UCWABND.....	5
<b>Copy Members.....</b>	<b>5</b>
New and Modified Copy Members.....	5
CPWSWEBS (new).....	5
CPWSFLT (new).....	5
CPWSTOKN (new).....	5
CPPDFLT (new).....	5
CPPDTOKN (new).....	5
PS002 (new).....	5
PS002I01 (new) .....	6
PS002O01 (new).....	6
CPPDEDHC .....	6
CPWSPTRW .....	6
CPWSTRIF.....	6
<b>Bind Members.....</b>	<b>6</b>
New and Modified Binds.....	6
PPDELTS (new) .....	6
<b>Web Artifacts .....</b>	<b>6</b>
PPSPROV .....	6
PPSbasicSOAP11provider.xml.....	7
PS002.wsdl .....	7
PS002.wsbind .....	7
<b>Control Table Changes.....</b>	<b>8</b>
System Messages Table (PPPMSG) .....	8
UCOCFN .....	8
UCOPGM .....	8
<b>Appendix A - Overview of CICS Web Services .....</b>	<b>9</b>
wsdl.....	9
RADz.....	9

<b>Appendix B - Overview of PPS Web Services Security .....</b>	<b>10</b>
Server Level Security .....	10
Application Level Security .....	10
<b>Appendix C - Tehnical Note on Unbounded Arrays.....</b>	<b>11</b>

## **Introduction**

### **Service Request 82205**

Service Request 82205 asks that a service interface be developed between the Payroll/Personnel System (PPS) online time reporting system and any time/attendance collection application capable of transmitting data over a network using common protocols.

Service Request 82205 provides the following background information:

In 2007, the ESI Planning Committee requested that the Payroll/Personnel Services unit design and develop a generic Payroll/Personnel System (PPS) web service to support various time collection applications in use throughout the University of California system.

The web service was to provide the ability to process a time transaction, execute existing PPS edits, post error-free time data to PPS, and return appropriate messages back to the time collection application.

The scope of the project is limited to the intermediary service that would provide a link between PPS time reporting components and a time collection application.

## **Overview of CICS Web Services**

See Appendix A.

## **Overview of PPS Web Services Security**

Although this release does not include a web application to invoke the web service, it is assumed that campuses will be deploying web applications for that purpose.

See Appendix B for information on PPS Web Security.

## **Overview of PPS Modifications**

A CICS web service interface will be developed based on the existing CICS time reporting functions. New program PS002 will be written to invoke existing CICS programs, starting with UCROUTER. Existing programs will be modified to recognize web service calls and handle them appropriately, but existing CICS functionality will be unchanged.

Due to technical issues when the web service requests contain a large number of transactions, a limit of 100 transactions per request will be imposed. If the technical issues are resolved in the future, the limit may be removed.

### **PPS CICS Infrastructure**

The existing CICS application infrastructure will be modified to allow it to be run in a web service context. All existing logic will be untouched, but logic will be added to support web services.

### **Web Services Programs**

New CICS web service programs will be written to handle the SOAP messages from the web clients.

### **Web Artifacts**

The web service wsdl will be generated by RADz as part of the development process. Other web service related items will be written to support the PPS services.

## Tokens

In CICS, tokens created and managed by IBM supplied CICS program UC0\$WBST can be used to track the "state" of an operation which is not related to a specific terminal. A token is a randomly generated number used to identify an active "session". Session-related data (eg. user ID) can be saved in main storage with the token as the key. This use of tokens will be used in later PPS web service projects.

For the time reporting service, however, tokens will be used for a different purpose. There are features of the PPS application which rely on a unique terminal ID to track the user's session. For example data is stored into temporary storage with the terminal ID as part of the key. Since web services are not associated with terminals, the token will be used in place of the terminal ID, since it uniquely identifies a session.

The time reporting service will be developed to accept a token as an input field, so that, in the future, it can take advantage of tokens generated by future PPS web services. Because those services do not yet exist, the only valid value for the input token at this time is 9999, indicating that no token is being passed to the service.

## IDTC/EDHC Process

Minor modifications will be made to the existing IDTC/EDHC processing programs to support web services. Existing functionality will remain unchanged.

## Programs

### New and Modified Programs

#### UC0\$WBST (new)

UC0\$WBST is a clone of an IBM-supplied assembler language program called DFH\$WBST used to "maintain state" in a CICS environment. This is done by creating a token and storing user data in main storage to be retrieved by a later task..

A modified version of DFH\$WBST will be used for PPS web services as follows:

- o UC0\$WBST will call new program PPDELTS when deleting a token. PPDELTS will delete all temporary storage queues associated with the token.
- o The token will be converted to a 4-byte value which will replace the terminal ID in temp storage queue names.

#### PPWEBSEC (new)

PPWEBSEC will be the web security interface program. It will be invoked as part of the SOAP header processing. It will validate and process the user ID, password and/or token passed in the SOAP message from the web application. Processing will be as follows:

- o Obtain the SOAP body from the DFHWS-BODY container.
- o The program obtains the APPLID of the CICS region. In Working-Storage, there's a list of CICS regions which do NOT require SSL, which can be modified as needed for testing. This should be used only for test regions, and *never for production regions*. If the current region is not in this list, then the program will perform certificate checking to ensure that the requestor did not attempt to enter via a non-SSL port. If the certificate isn't valid, a SOAP fault will be generated.
- o Call PPXMLPRS program to parse the XML and extract the user ID, password and/or token.
- o If the token is equal to 9999, then a valid user ID and password must be included. The program will verify the user ID/password combination. If it is invalid, a SOAP fault will be generated.

- If the token is not equal to 9999, user ID and password are ignored. The token will be passed to program UC0\$WBST which will obtain the user ID from the stored token data.
- Then the program will place the user ID into the DFHWS-USERID container. This causes the CICS task to run under that user ID.

### **PPXMLPRS (new)**

PPXMLPRS will be the PPS XML parser. It will search the SOAP message for user ID, password and/or token tags and return them to the calling program. It will be called by PPWEBSEC. Processing will be as follows:

- Parse the XML data passed in the Commarea.
- Search for the following XML tags:
  - webserv\_user\_id,
  - webserv\_password
  - webserv\_user\_token\_in.
- When found, extract the data and store it in the commarea.

### **PPDELTS (new)**

PPDELTS will delete all PPS temporary storage queues associated with the token passed in the Commarea. Processing will be as follows:

- Read the UCOSI table to get the names of each PPS subsystem. If subsystem names are not available from UCOSI for any reason, perform subsystem processing for the "UC" and "EU" subsystems.
- For each subsystem:
  - Attempt to read the "audit" temporary storage queue for the subsystem and the user token. The audit queue contains the names of all other temporary storage queues for that subsystem and token.
  - If the audit queue was found, delete each temporary storage queue identified in the audit queue.

### **UCROUTER**

UCROUTER is the main driver for all PPS CICS processes. It will be modified to support web services with no changes to existing CICS functionality. UCROUTER will interrogate the CPWSWEBS-WEB-SWITCH to determine whether certain logic should be executed or excluded. Major modifications will be as follows:

- A new external defined by copylib member CPWSWEBS will be added to pass web-service related information to UCROUTER.
- For a web service call, a GOBACK is used instead of EXEC CICS RETURN because UCROUTER is invoked by a CALL for web services.
- For a web service call, logic to format, send and receive CICS maps will be bypassed.
- For a web service call, a special logical program name will be used for interpreting the header and footer map areas. The logical program is WINTERP-HD-FT. This will allow processing of pay cycle data from the "main menu" process, which is not included in the regular main menu footer map.
- At this time, web services will not support interactive PAN processing through UCROUTER. Therefore, for web services, interactive PAN processing will be bypassed.
- For a web service call, processing of the "help" temporary storage queue will be bypassed.

### **PPGETTIM**

PPGETTIM retrieves data from the PPPTIM table for online use, specifically for the time reporting roster screen. It will be modified as follows:

- A new mode will be added to select data for a single employee's distribution (rather than the typical multi-row roster). The program will detect the single-row mode when passed an employee ID in the CPWSPTRW interface and will set WS-SINGLE-ROW-SELECTION in working-storage.
- When the request is to count rows, PPGETTIM will count the PPPTIM rows for just the one employee.
- When the request is to select rows, PPGETTIM will fetch from a new cursor, SINGLE-ROW-CURS which obtains PPPTIM data for the employee selected.

### **PPWEDHC**

PPWEDHC is the entry screen program for the time reporting function, for entry of detail time data, after the roster is specified on the IDTC screen. For the web service it will be called by UCROUTER using data passed from PS002.

It will be modified to improve the processing of error messages in support of the web service.

### **PPWIDTC**

PPWIDTC is the first of the two CICS screen programs in the time reporting function. Typically the user fills in pay cycle data and is then routed to the EDHC screen for detailed entry. For the web service, it will be called by UCROUTER using data passed from PS002. It will be modified as follows:

- New external CPWSWEBS-WEB-INFO (copylib member CPWSWEBS) will be added so that the program can detect when it's being used in a web service context (CPWSWEBS-WEB-CALL)
- The code to edit the type of request (3200-EDIT-REQUEST-TYPE) will be bypassed when it's a web service call.
- The code which handles the "no data entered" condition will be bypassed when it's a web service call.
- For a web service call, the next function will always be EDHC (in the CICS interface, the IPIC function can also be invoked).
- Since the valid pay cycle will be passed in the web service interface, the code which processes selecting a pay cycle will be bypassed for a web service call.

### **PS002 (new)**

PS002 will be the time reporting web service driver program. The shell for this program will be initially generated by RADz from PS002.wsdl (see Appendix C). Then the shell program will be modified for the time reporting service.

Processing will be as follows:

- Obtain the CICS containers which contain the parsed XML which was sent from the web application. For the recurring data structure a GETMAIN will be done to obtain storage for the array.
- If no token was passed, generate a temporary token using the user ID which was passed so that it can be used as the terminal ID for temporary storage queue names.
- Set the "terminal" ID from the token.
- Make an initial call to UCROUTER to simulate the main menu process.
- Call UCROUTER with a request for the WDTC function. This function is analogous to IDTC, but the store and restore logical functions are dummied out in the UCOPGM table to prevent data passed to UCROUTER from being overridden by the restore process.
- For each distribution in the XML array, the following will be performed:

- Input key data will be moved to the CPWSPTRW interface, and UCROUTER will be called to simulate the entry of data into the IDTC screen. This will cause PPGETTIM to be called to obtain the PPPTIM data for the employee and distribution passed.
- The detail data will be moved to the external map area for the EDHC screen. UCROUTER will be called to simulate the edit and update of the time data.
- Error message data will be moved to the output XML interface area.
- UCROUTER will be called again to simulate backing out to the IDTC screen.
- After all distributions have been processed, UCROUTER will be called one final time to simulate exiting the application.
- If a new token was generated for this session, destroy it.
- All message data will be stored in the return request area which will be translated to XML.
- The return data is written to the appropriate CICS containers.

### **UCWABND**

UCWABND is the abend handling program for the online applications system. It will be modified to bypass sending a CICS map when processing a web service.

## **Copy Members**

### **New and Modified Copy Members**

#### **CPWSWEBS (new)**

CPWSWEBS will define a new external area for passing data between the web services related programs.

#### **CPWSFLT (new)**

CPWSFLT will be the work area for processing SOAP faults. It's associated with procedure division copylib member CPPDFLT.

#### **CPWSTOKN (new)**

CPWSTOKEN will be the work area for dealing with tokens. It's associated with procedure division copylib member CPPDTOKN.

#### **CPPDFLT (new)**

CPPDFLT contains code for processing a SOAP fault. It's associated with Working-Storage copylib member CPWSFLT.

#### **CPPDTOKN (new)**

CPPDTOKN contains code for processing tokens. It's associated with Working-Storage copylib member CPWSTOKN.

#### **PS002 (new)**

PS002 will define the input and output structure for the web service. It will be used only to generate the wsdl and will not be copied into any Cobol program. Any changes to the structure require that the wsdl be re-generated. (see Appendix C)

### **PS002I01 (new)**

PS002I01 will be the web service input data structure. It will be generated by RADz from PS002.wsdl at the same time as the PS002 shell program is generated. This copylib should never be modified. (see Appendix C)

### **PS002O01 (new)**

PS002O01 will be the web service output data structure. It will be generated by RADz from PS002.wsdl at the same time as the PS002 shell program is generated. This copylib should never be modified. (see Appendix C)

### **CPPDEDHC**

CPPDEDHC is the procedure division logic related to FAU for the EDHC screen. It will be modified to obtain the FAU from the data passed from the web service instead of using the screen input area.

### **CPWSPTRW**

CPWSPTRW is the common work area for the time entry CICS functions. The following two fields will be added to facilitate passing data between the web service and the time entry programs (filler will be used, so no change to the work area structure will occur):

- CPWSPTRW-DIST-NUM
- CPWSPTRW-BW-FISC-END

### **CPWSTRIF**

CPWSTRIF is the interface area for the PPGETTIM program. It will be modified to support a new mode for PPGETTIM which will cause the program to retrieve data for a single employee rather than a roster.

## **Bind Members**

### **New and Modified Binds**

#### **PPDELTS (new)**

This bind member will contain the package bind for PPDELTS.

## **Web Artifacts**

### **PPSPROV**

PPSPROV is the web service provider pipeline used for PPS web services. This will be defined to CICS using RDO. The RDO statements to define the pipeline are shown below.

```
DEFINE PIPELINE(PPSPROV) GROUP(*GROUPID*)
    STATUS(ENABLED) RESPWAIT(DEFT)
    CONFIGFILE(*PIPELINEDIR*/PPSbasicSOAP11provider.xml)
    SHELF(*SHELFDIR*) WSDIR(*PICKUPDIR*)
```

Explanations of some of the parameters follow:

- CONFIGFILE(\*PIPELINEDIR\*/PPSbasicSOAP11provider.xml) - This is the configuration file for the pipeline. \*PIPELINEDIR\* will be replaced by the HFS pipeline directory defined by systems for the CICS region.
- SHELF(\*SHELFDIR\*) - This is used by CICS for configuring web services.
- WSDIR(\*PICKUPDIR\*) - This is where CICS will look for new wsbind files when a pipeline scan is performed.

### **PPSbasicSOAP11provider.xml**

PPSbasicSOAP11provider.xml is a customized version of the standard pipeline configuration file, basicSOAP11provider.xml. It was customized to include the header program PPWEBSEC which is the PPS web service security module. The file will be loaded into the pipeline directory defined for the PPSPROV pipeline.

This file is shown below with the customization highlighted.

#### **PPSbasicSOAP11provider.xml**

```
<?xml version="1.0" encoding="EBCDIC-CP-US"?>
<provider_pipeline xmlns="http://www.ibm.com/software/http/cics/pipeline">
  <service>
    <terminal_handler>
    <cics_SOAP_1.1_handler>
    <headerprogram>
      <program_name>PPWEBSEC</program_name>
      <namespace>http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-
1.0.xsd</namespace>
      <localname>Security</localname>
      <mandatory>true</mandatory>
    </headerprogram>
    </cics_SOAP_1.1_handler>
  </terminal_handler>
</service>
<apphandler>DFHPITP</apphandler>
</provider_pipeline>
```

### **PS002.wsdl**

The wsdl is the true interface for the web service, although it is not actually used by the service itself. The wsdl describes the service so that web application developers can successfully call the web service. PS002.wsdl will be generated in RADz with a "bottom-up" approach, using the PS002 copylib member as input. (see Appendix C). The wsdl is then used to generate the wsbind file.

### **PS002.wsbind**

The wsbind file defines the web service to CICS. The wsbind file is generated from the wsdl. It can be generated either by web service tools in RADz or by running batch utility DFHWS2LS. Once generated, PS002.wsbind will be moved to the web service "pickup directory" defined for the PPSPROV pipeline.

## Control Table Changes

### System Messages Table (PPPMSG)

Error Messages 36-125 and 36-127 will have the message text change per the Service Request. The new values are

36-125 PTO BONUS HOURS REPORTED ARE GREATER THAN PTO BALANCE

36-127 PTO BONUS HOURS PAID ARE LESS THAN PTO BALANCE

### UC0CFN

UC0CFN is an infrastructure table in the Process Control Database (PCD) which contains all of the screen function codes and their associated programs.

A new function, WDTC, will be added to this table which will be used instead of IDTC for the web service call. This will allow for function-specific overrides to be added to the UC0PGM table.

### UC0PGM

UC0PGM is an infrastructure table in the Process Control Database (PCD) which contains the mapping of logical program functions to actual programs for the PPS CICS applications.

A record will be added to it for processing a new "logical function" called WINTERP-HD-FT, for processing the simulated footer map for the web service. This table entry will be set to invoke program PPAPTFIH which processes payroll cycle data in addition to employee key data.

UCROUTER will contain modifications to use INTERP-HD-FW instead of INTERP-HD-FT for specified web services.

Program overrides will be used for the WDTC function for the following "logical functions"

- RESTORE-ITER
- RESTORE-NEST
- STORE-ITER
- STORE-NEST

It is necessary to bypass these processes for the web service, so they will contain program name UCAPDUMM in this table which will cause no action to be taken for those logical functions.

## **Appendix A - Overview of CICS Web Services**

CICS web services allow web applications to invoke CICS programs. The CICS programs being called must not attempt to send or receive CICS maps, but otherwise can perform the same operations they do when invoked within CICS. Data is passed from the web application to CICS in XML format which is then translated to a Cobol data structure and processed normally. The results are translated back to XML before sending them back to the requesting web application.

It's important to remember that a web service is not in itself a web application. It's contained wholly within CICS and provides access to CICS logic from a web application via a Web Service Request.

### **wsdl**

Data is passed between the web application and CICS in XML format. The format is defined in a file called a wsdl which web application developers use to correctly format the data for the service. The wsdl is generated as part of the web service development process.

### **RADz**

RADz is an IBM product which contains tools to assist in building CICS web services. Starting with a Cobol data structure, RADz will build the wsdl and a Cobol interface program to process the XML.

## **Appendix B - Overview of PPS Web Services Security**

Although this release does not include a web application to invoke the web service, it is assumed that campuses will be deploying web applications for that purpose. This section addresses the approach to security for web applications invoking PPS web services.

Two levels of authentication and authorization are required for Web Services. At the server level the CICS region providing Web Services must authenticate the server where the requesting Web application runs. At the application authorizations level the Web service must determine if the logged on user is authorized to use the Web Service.

### **Server Level Security**

In order to validate the web application server which is making a request to the CICS web service, an SSL certificate must accompany the request. As part of the installation, a "Certificate Signing Request" (CSR) must be generated by the web application server and sent to the mainframe Certificate Authority which will generate a Certificate which is sent back to the web server administrator to be installed on the web application server. When the certificate is generated for the mainframe, a RACF user ID is associated with the certificate.

Server level authentication and authorization is implemented by applying CICS resource security to CICS transaction CPIH. All CICS Web Service requests are run under transaction CPIH. The user ID associated with the certificate must be permitted to use transaction CPIH in the CICS region receiving the request. This allows RACF permissions to control which Web application servers can make requests to a CICS region.

### **Application Level Security**

Once the web application server authorization has been passed, the user's authorization to use the web service must be validated. This is done by forcing the CICS transaction to run under the logged in user's ID and then using the existing RACF and ARSM rules for authorizing access to the application.

## Appendix C - Tehnical Note on Unbounded Arrays

The PS002 web services requires an unbounded array to contain a large number of possible time input transactions. The RADz tools do not successfully implement unbounded array. Therefore, a multi-step approach will be used for generating the web service programs:

1. Copylib member PS002 describes the input and output structures for the web service. This copylib member will not be copied into any programs. It will be exclusively used to generate the WSDL for the service using RADz (bottom-up approach).
2. Once the WSDL is generated, it must be manually modified so that the arrays are unbounded.
3. The WSDL is then used in RADz to generate the web service shell program PS002 (top-down approach). This process will also generate the interface copylib members PS002I01 & PS002O01.

**Important:** Note that any changes to the input and output structure will cause copylib member PS002 to change. This will require that the PS002.wsdl be re-generated and modified again as in steps 1 & 2 above. After that, step 3 must be repeated to re-generate PS002I01 & PS002O01, but the PS002 shell program which is generated by this process should not be used. Instead, the PS002 Cobol program should be maintained manually.