



**Masoud Aghadavoodi Jolfaei**

November 18, 2013 12 minute read

## ABAP Channels Part 1: WebSocket Communication Using ABAP Push Channels

[Follow](#)

[RSS feed](#)

7 Likes 22,075 Views 57 Comments

### Introduction

For introduction to ABAP Channels please refer to [Introduction to ABAP Channels](#). For tangible examples in the system see also the blog [ABAP Channels Examples](#). See also [FAQ – ABAP Channels](#).

This blog focuses on the WebSocket integration in ABAP, also known as **ABAP Push Channel** (APC).

For live demonstration and step-by-step implementation take a look at [ABAP Push Channel Video](#).

### Integration of WebSocket into ABAP engine

In order to pass an event from the ABAP engine to an HTML5 based user agent we decided to integrate the WebSocket protocol into the ABAP engine. WebSocket provides a bi-directional communication channel over a

TCP/IP socket. It is designed to be implemented in web browsers and web servers, but in general it can be used by any client or server application. The WebSocket JavaScript API is standardized by the W3C, whereas the WebSocket protocol (RFC 6455) is standardized by the IETF. The WebSocket protocol is supported by many new browsers, e.g. Chrome 14, Firefox 7.0, IE 10 and Safari on iOS6 (and higher).

The implementation of the WebSocket protocol (RFC 6455) in ABAP is called *ABAP Push Channel* (APC). The APC communication is stateless per default, i.e. each WebSocket message leads to the creation of an individual ABAP session for execution of the respective method. After termination of the method the session will be released. In contrast to the existing request-response-based communication protocols, e.g. Remote Function Call (RFC) and HTTP(S), APC is a message-based protocol, i.e. per default the execution of a message does not lead to a response. However on top of the messaging protocols various communication patterns can be built, also a request-response-based communication. This is possible because the APC session has been implemented as a new session type..

It is important to have an understanding for the WebSocket events and methods before starting to explain how the WebSocket protocol has been integrated into the ABAP engine. The following simple sample code snippet shows how to use the JavaScript WebSockets interface (the essential events and methods are in bold format):

WebSocket JavaScript example code	
<pre>// open a WebSocket connection  var ws = new <b>WebSocket</b>("ws[s]://host.domain:port/path?query_string");  ws.<b>onopen</b> = function() {     // WebSocket is connected, send data using send()      ws.<b>send</b>("Sending Message ....");      alert("Message is sent..."); };  ws.<b>onmessage</b> = function (msg)</pre>	

#### WebSocket JavaScript example code

```
{  
  
    // process received Message  
  
    alert(msg.data);  
  
  
    // optionally close connection  
    ws.close();  
  
};  
  
ws.onclose = function()  
{  
  
    // WebSocket is closed.  
  
    alert("Connection is closed...");  
  
};
```

In APC framework the involved events and methods are very similar. A simple APC class handler would have the following events and methods.

In order to support WebSocket specification for subprotocols according to RFC 6455 has the APC server interface IF\_APC\_WS\_EXTENSION been refactored and the new interface **IF\_APC\_WSP\_EXTENSION with 740 SP08** has been established.

#### APC example code (before 740 SP08)

CLASS YCL\_APC\_WS\_EXT\_YEXAMPLE IMPLEMENTATION.

METHOD if\_apc\_ws\_extension~on\_start( i\_context type ref to if\_apc\_ws\_context ).

” WebSocket connection has been established

ENDMETHOD.

METHOD if\_apc\_ws\_extension~**on\_message**( i\_message type ref to if\_apc\_ws\_message ).

” Message has been received

TRY.

“ Echo the message on WebSocket connection

data(lo\_message\_manager) = i\_message->get\_context( )->get\_message\_manager( ).

lo\_message = lo\_message\_manager->create\_message( ).

lo\_message->set\_text( i\_message->get\_text( ) ).

” Send message

lo\_message\_manager->**send**( lo\_message ).

CATCH cx\_apc\_error INTO lx\_apc\_error.

...

ENDTRY.

ENDMETHOD.

METHOD if\_apc\_ws\_extension~**on\_close**.

” WebSocket is closed

ENDMETHOD.

ENDCLASS.

The new version of APC class handler would have the following events and methods, which also supports the Push Channel Protocol (PCP) as subprotocol.

APC example code (from 740 SP08)

CLASS YCL\_APC\_WSP\_EXT\_YEXAMPLE IMPLEMENTATION.

METHOD if\_apc\_wsp\_extension~on\_start(  
i\_context type ref to if\_apc\_wsp\_server\_context  
i\_message\_manager type ref to if\_apc\_wsp\_message\_manager ).

” WebSocket connection has been established

ENDMETHOD.

METHOD if\_apc\_wsp\_extension~on\_message(  
i\_message type ref to if\_apc\_wsp\_message  
i\_message\_manager type ref to if\_apc\_wsp\_message\_manager  
i\_context type ref to if\_apc\_wsp\_server\_context ).

” Message has been received

TRY.

“ Echo the message on WebSocket connection

lo\_message = i\_message\_manager->create\_message( ).

lo\_message->set\_text( i\_message->get\_text( ) ).

” Send message

i\_message\_manager->send( lo\_message ).

CATCH cx\_apc\_error INTO lx\_apc\_error.

...

# APC example code (from 740 SP08)

ENDTRY.

ENDMETHOD.

METHOD if\_apc\_wsp\_extension~on\_close(

i\_reason type string

i\_code type type i

i\_context\_base type ref to if\_apc\_wsp\_server\_context\_base ).

" WebSocket is closed

ENDMETHOD.

ENDCLASS.

The following diagram (figure 2.0) shows the interaction model of WebSocket client, e.g. HTML5-enabled browser, and APC framework in the ABAP engine of the NetWeaver application server.

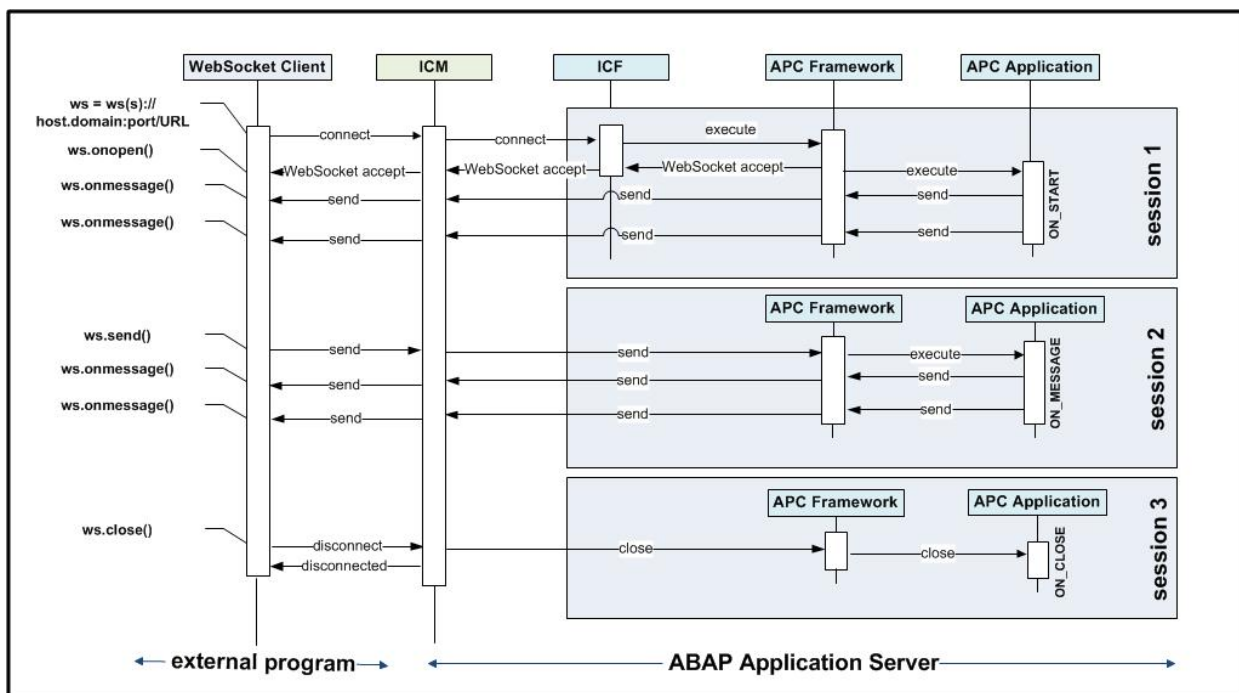


Figure 2.0: WebSocket/APC interaction model in ABAP

The main advantage of the integration model is based on the fact that WebSocket connections are administrated by an ICM process, i.e. the end-point of a connection is the ICM process and not the ABAP session. This design helps to reduce resource consumption during inactivity phases of WebSocket communication. Furthermore APC applications are stateless per default. This means that each received message is handled in a dedicated session. The application can preserve its state information either in shared object areas or database tables.

## Creating an ABAP Push Channel (APC) application

This section contains a step by step description of the creation of an ABAP Push Channel application.

Before starting with development of an APC application we should check the prerequisites for the WebSocket environment in ABAP. From support package 5 upwards the WebSocket configuration is active per default. The WebSocket connection uses the same ports as the maintained HTTP and HTTPS ports and there is no explicit entry for **“WEBSOCKET”** and **“WEBSOCKETS”** services in service entries of transaction **SMICM**. This is different in lower releases. In NetWeaver 740 support package 2 up to support package 4, the WebSocket port parameters were maintained actively. This can be done in two different ways:

- Manually and temporarily in the transaction **SMICM** (via menu entry **“Goto”** -> **“Service”** and again in menu **“Service”** -> **“Create”**) you can choose a proper port number for the field **“New Service Port”**, e.g. 50090 or 44390 for secure or non-secure WebSocket (WS), and then in the field **“Protocol”** enter **“WEBSOCKET”** (non-secure WebSocket) or **“WEBSOCKETS”** (secure WebSocket).
- Alternatively, enabling ABAP to use WebSockets can be performed by setting ICM profile parameters. In the profile of the instances supporting WebSockets, some parameters must be added. Example:

```
icm/server_port_101 = PROT=WEBSOCKET,PORT=0,TIMEOUT=-1  
icm/server_port_102 = PROT=WEBSOCKETS,PORT=0,TIMEOUT=-1
```

assuming that 101 and 102 are not yet used by other server\_ports.

Port number is always 0. Timeout is always -1.

An ABAP Push Channel (APC) application can be created in two ways.

- In transaction **SE80** using the context menu entry (**“Right-Click on top-level package object”**) **“Create”** -> **“Connectivity”** -> **“ABAP Push Channel”** (see figure 2.1):

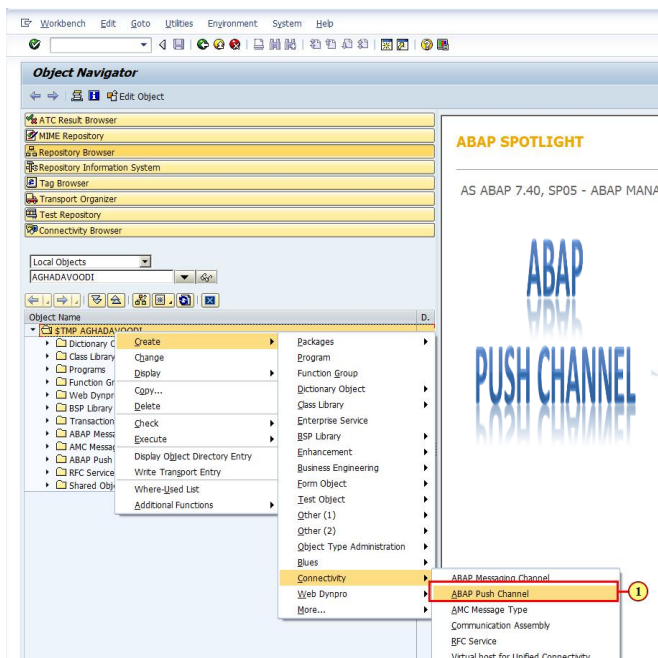


Figure 2.1.

- As of release 740 SP5: In the transaction **SAPC** using the context menu entry ("Right-Click") on the "**ABAP Push Channel**" entry and selecting "**Create**" (see figure 2.2):

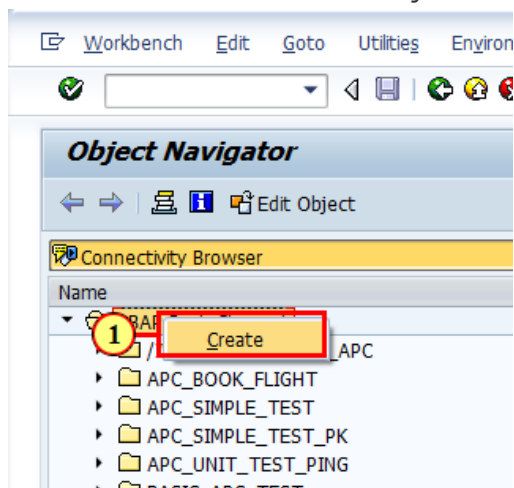


Figure 2.2.

In the next step, the name of the APC application has to be entered and the popup screens have to be confirmed (press "**Enter**"). In this example "**YAPC\_WS\_TEST**" is used as application name (see figure 2.3):

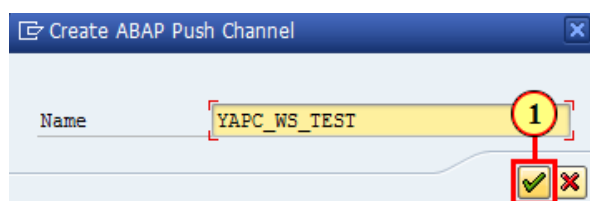


Figure 2.3.



Also choose an appropriate package entry, in this case as “**Local Object**” (see figure 2.4):

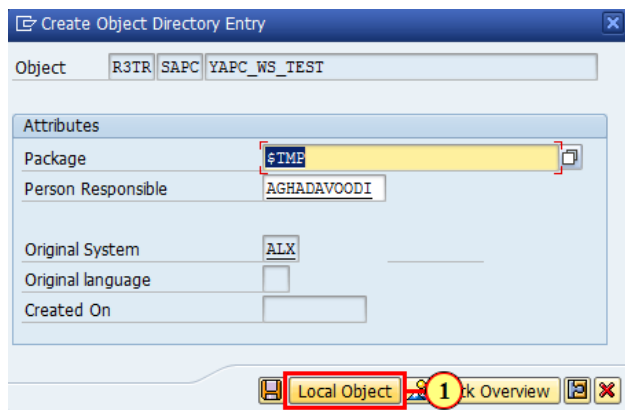


Figure 2.4.

Additionally, the field “**Description**” has to be maintained (see figure 2.5):

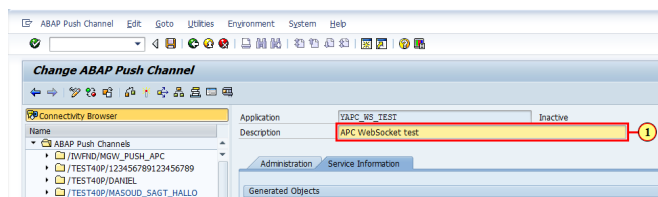


Figure 2.5.

Next step: save the application (see figure 2.6):

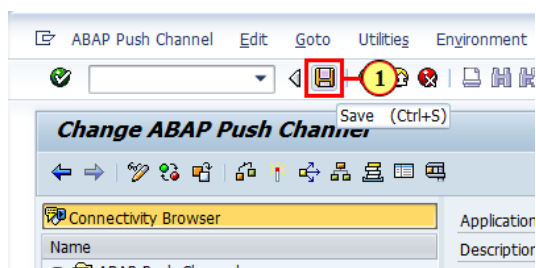


Figure 2.6.

Now the associated design time entities, i.e. in the transaction **SICF** service path `/sap/bc/apc/<name space>/<application name>` and the corresponding APC application class, have to be generated (see figure 2.7):

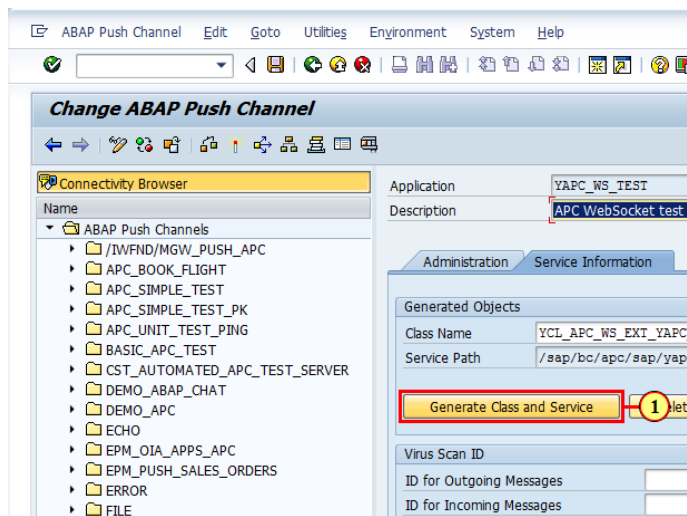


Figure 2.7.

Confirm the generation step.

In the next steps the basic implementation of the APC extension class will be initiated. To do so, double-click on the class name, here `YCL_APC_WS_EXT_YAPC_WS_TEST` (see figure 2.8):

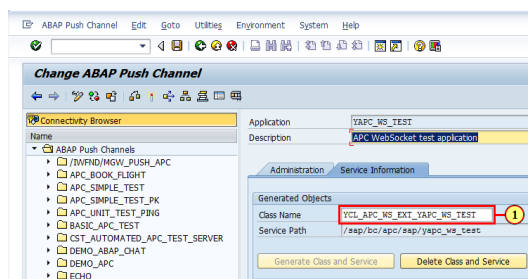


Figure 2.8.

The before-mentioned action leads to the display of the class `YCL_APC_WS_EXT_YAPC_WS_TEST` in the class builder environment (see figure 2.9):

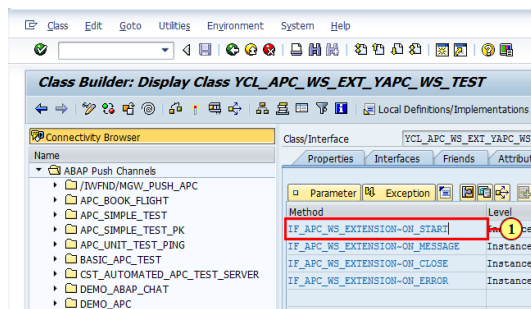


Figure 2.9.

Change the display mode to *change*, i.e. click “Display” <-> “Change” (see figure 2.10).

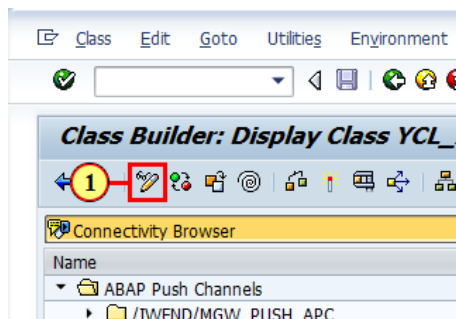


Figure 2.10.

The required actions are the implementation of the methods **ON\_START** and **ON\_MESSAGE**. The **ON\_START** method is executed as soon as the WebSocket connection setup phase is accomplished successfully. The **ON\_MESSAGE** method is executed when receiving a message. As the extension class is inherited from an abstract class to implement the **ON\_START** and **ON\_MESSAGE** methods these methods have to be redefined. Optionally also the method **ON\_CLOSE** and **ON\_ERROR** can be redefined, i.e. implemented, as well. To do so, choose the respective method, i.e. **ON\_START**, and press “Redefine” (see figure 2.11):

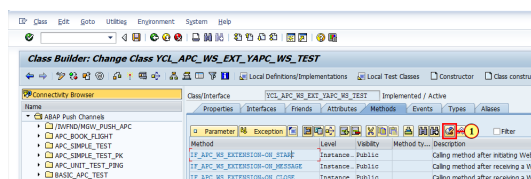


Figure 2.11.

In this example the only action which is processed during **ON\_START** execution is to send a text message to the WebSocket client, after inserting the following code in the method **ON\_START**:

The example coding for 740 with support package less than SP08 is:

**METHOD if\_apc\_ws\_extension~on\_start.**

TRY.

\* send the message on WebSocket connection

```
DATA(lo_message_manager) = i_context->get_message_manager( ).
```

```
DATA(lo_message) = lo_message_manager->create_message( ).
```

```
lo_message->set_text( |{ sy-mandt }/{ sy-uname }: ON_START has been successfully executed !| ).
```

```
lo_message_manager->send( lo_message ).
```

```
CATCH cx_apc_error INTO DATA(lx_apc_error).
```

```
MESSAGE lx_apc_error->get_text( ) TYPE 'E'.
```

ENDTRY.

**ENDMETHOD.**

And the coding for 740 SP08 and later is:

**METHOD if\_apc\_ws\_extension~on\_start.**

TRY.

\* send the message on WebSocket connection

```
DATA(lo_message) = i_message_manager->create_message( ).
```

```
lo_message->set_text( |{ sy-mandt }/{ sy-uname }: ON_START has been successfully executed !| ).
```

```
i_message_manager->send( lo_message ).
```

```
CATCH cx_apc_error INTO DATA(lx_apc_error).
```

```
MESSAGE lx_apc_error->get_text( ) TYPE 'E'.
```

ENDTRY.

ENDMETHOD.

the method contains the code below (see figure 2.12):

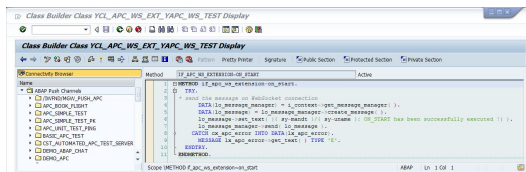


Figure 2.12.

Save the method (click “Save”) in the “Class Builder” (see figure 2.13):

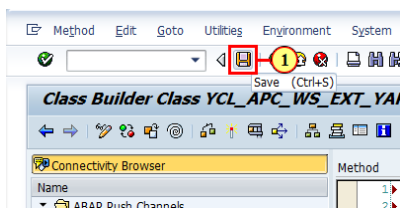


Figure 2.13.

and leave the method (click “Back”), see figure 2.14:

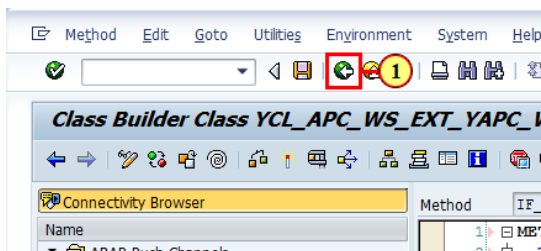


Figure 2.14.

In the **ON\_MESSAGE** method both a message text and the received message are sent to the WebSocket client, after inserting the following code in the **ON\_MESSAGE** method:

The example coding for 740 with support package less than SP08 is:

#### **METHOD if\_apc\_ws\_extension~on\_message.**

TRY.

\* retrieve the text message

```
DATA(lv_text) = i_message->get_text( ).
```

\* create the message manager and message object

```
DATA(lo_context) = i_message->get_context( ).
```

```
DATA(lo_message_manager) = lo_context->get_message_manager( ).
```

```
DATA(lo_message) = lo_message_manager->create_message( ).
```

\* send 1st message

```
lo_message->set_text( |{ sy-mandt }/{ sy-uname }: ON_MESSAGE has been successfully executed  
!| ).
```

```
lo_message_manager->send( lo_message ).
```

\* send 2nd message, i.e. echo the incoming message

```
lo_message->set_text( lv_text ).
```

```
lo_message_manager->send( lo_message ).
```

```
CATCH cx_apc_error INTO DATA(lx_apc_error).
```

```
MESSAGE lx_apc_error->get_text( ) TYPE 'E'.
```

ENDTRY.

ENDMETHOD.

And for 740 SP08 and later release is:

**METHOD if\_apc\_ws\_extension~on\_message.**

TRY.

\* retrieve the text message

DATA(lv\_text) = i\_message->get\_text( ).

\* create the message object

DATA(lo\_message) = lo\_message\_manager->create\_message( ).

\* send 1st message

lo\_message->set\_text( |{ sy-mandt }/{ sy-uname }: ON\_MESSAGE has been successfully executed  
!| ).

i\_message\_manager->send( lo\_message ).

\* send 2nd message, i.e. echo the incoming message

lo\_message->set\_text( lv\_text ).

i\_message\_manager->send( lo\_message ).

CATCH cx\_apc\_error INTO DATA(lx\_apc\_error).

MESSAGE lx\_apc\_error->get\_text( ) TYPE 'E'.

ENDTRY.

ENDMETHOD.

The method contains the code below (see figure 2.15):

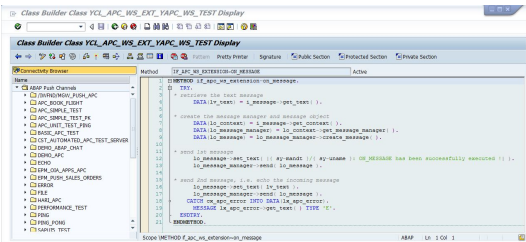


Figure 2.15.

Additionally save the method (click “Save”) in the “Class Builder” (transaction SE24), see figure 2.16:

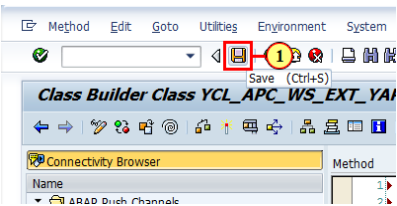


Figure 2.16.

and leave the method (click “Back”), see figure 2.17:

:

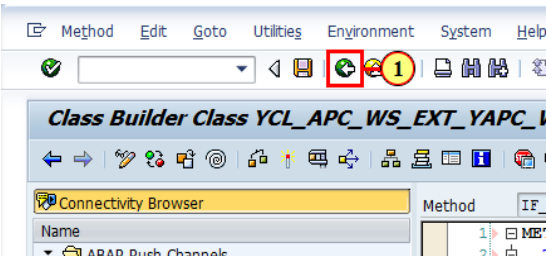


Figure 2.17.



Finally, activate the class (click “**Activate**”) in the “**Class Builder**”, see figure 2.18:

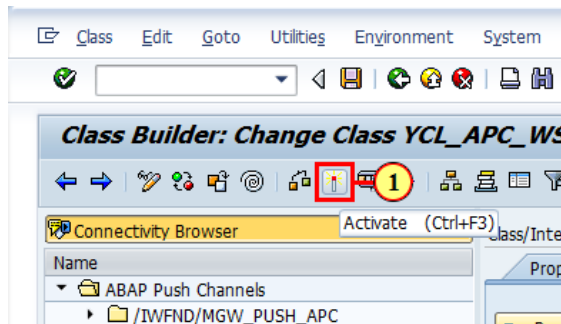


Figure 2.18.

and leave the “**Class Builder**” via *Back* (see figure 2.19):

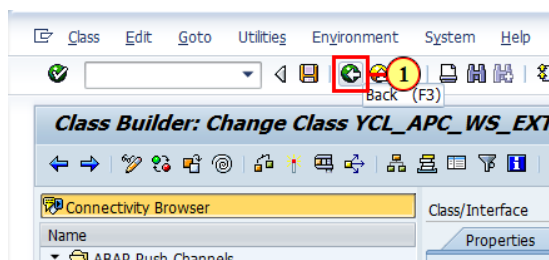


Figure 2.19.

Optionally, you can maintain the virus/content scan IDs for incoming and outgoing messages at any time (see figure 2.20):

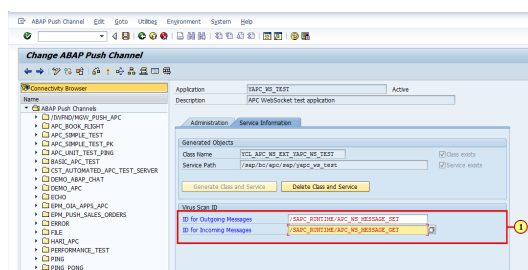


Figure 2.20.

Now the APC application has to be activated as well, to do so press **Activate**  (see figure 2.21):

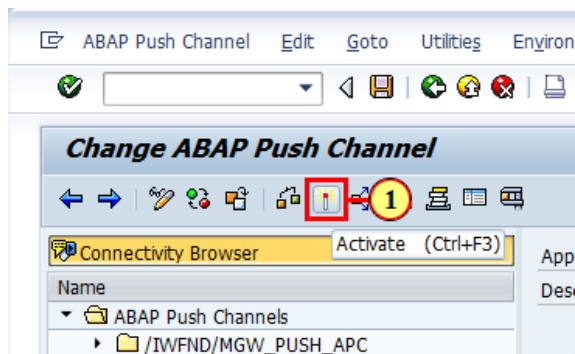



Figure 2.21.

To test the application just press the **Test** icon  (see figure 2.22):

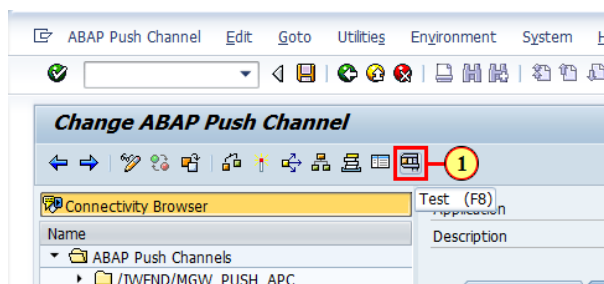


Figure 2.22.

This action launches the Web Dynpro application **WDR\_TEST\_APC** in a browser (see figure 2.23) which supports the WebSocket protocol (RFC 6455) (e.g. Chrome version  $\geq 16$ , Firefox version  $\geq 11$ , Internet Explorer version  $\geq 10$  and Safari on iOS version  $\geq 6$ ). In case of any issue it should be double checked that the Web Dynpro service path `"/sap/bc/webdynpro/sap/wdr_test_apc"` in the transaction **SICF** is active.

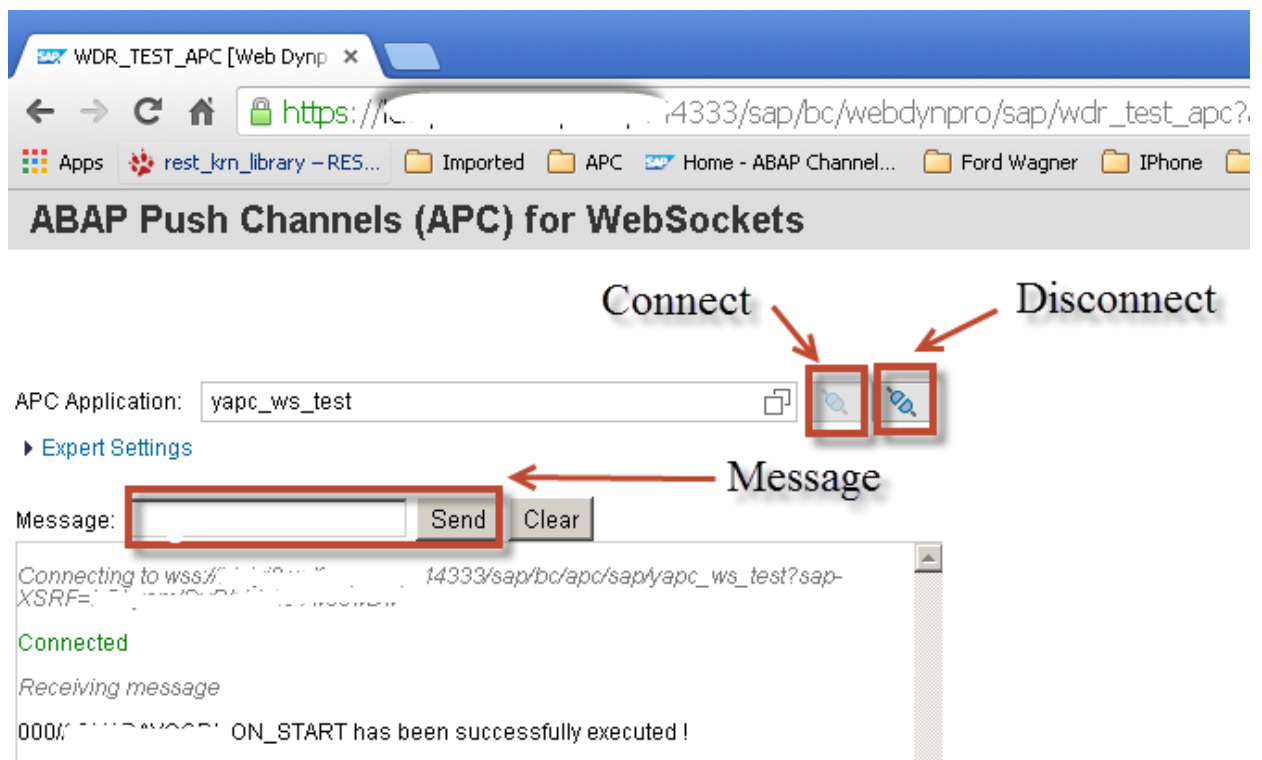


Figure 2.23.

**Limitation:** In the present version of APC framework is the size of WebSocket/APC messages exchanged between client and server limited to ~64 kbytes. Furthermore the WebSocket support in [Web Dispatcher](#) is released with the version 7.41 and for installation of a new version of Web Dispatcher please refer to SAP note [908097](#).

## APC Security

Each APC application has a path entry in the transaction **SICF**. With a new installation this entry is inactive per default. For an active usage of an APC application the associated APC path in **SICF** transaction has to be activated before. The path to an APC application is `/sap/bc/apc/<name space>/<application name>`. For each APC application you can maintain a dedicated virus/content scan ID for outgoing and incoming messages. Furthermore we recommend to use the existing escape methods in ABAP, e.g. the `ESCAPE` statement, to escape the WebSocket content that is exchanged between client and server. Finally we highly recommend to use the secure variant of WebSocket, i.e. **wss** instead of **ws**, for the communication.

For cross-origin-access to an APC application the APC framework also supports some aspects of the “The Web Origin Concept”, i.e. RFC 6454. In order to allow the access to APC application from an external server the white list maintenance via transaction **SAPC\_CROSS\_ORIGIN** has to be done accordingly.

## APC Supportability

With external breakpoints you can debug the execution of the **ON\_START**, **ON\_MESSAGE**, **ON\_CLOSE** and **ON\_ERROR** methods similarly to HTTP handler i.e. by setting breakpoints in the respective methods.

In transaction **SAPC** you can activate/deactivate the following supportability tools for a specific APC application:

be initiated:

- **Debugging:** choose menu bar the entry **“Utilities”** -> **“Debugging”** -> **“Activate Debugging”/“Deactivate Debugging”**
- **Runtime Analysis:** choose menu bar the entry **“Utilities”** -> **“Runtime Analysis”** -> **“Activate”/“Deactivate”**
- **Kernel Trace:** choose menu bar the entry **“Utilities”** -> **“Trace”** -> **“Reset Trace”/“Display Trace”/“Activate Trace”/“Deactivate Trace”**

The runtime analysis records which start with the prefix **“APC:<application path>”**, e.g. **“APC:/sap/bc/apc/sap/ping”** for a PING application, can be monitored in transaction **SAT**.

Furthermore, transaction **SMWS** shows the list of active WebSocket connections and their associated APC application on each application server.

## Conclusion and outlook

As you can see from the lines above, the ABAP Push Channel provides the infrastructure for WebSocket communication in ABAP. The present implementation of APC supports only the server side of communication based on stateless sessions, i.e. stateless APC communication similar to stateless HTTP communication. A stateful APC communication and also an APC client library are under development.

The next related blog to ABAP Channels is [ABAP Messaging Channel \(AMC\)](#). AMC blog describes the eventing framework for messages exchange between different ABAP sessions based on publish/subscribe channels.

Like   Alert Moderator

---

### Assigned tags

---

ABAP Connectivity | abap | channel | communication | event |

[View more...](#)

### Related Blogs

---

[Introduction to ABAP Channels](#)

By **Olga Dolinskaja** , Nov 27, 2014

[Real-time notifications and workflow using ABAP Push Channels \(websockets\) Part 1: Creating the APC and AMC](#)

By **Brad Pokroy** , Oct 16, 2014

[ABAP Channels Part 3: Collaboration Scenario Using ABAP Messaging and ABAP Push Channels](#)

By **Masoud Aghadavoodi Jolfaei** , Apr 14, 2014

### Related Questions

---

[Websockets oon HCP & Cloud Connector](#)

By **Carlos Zaburlin** , Aug 08, 2018

ABAP Push Channel is not supported in the selected project

By **Former Member** , Nov 03, 2016

ABAP Push Channel ...How to deal with data?

By **Amol Samte** , Oct 21, 2016

## 57 Comments

You must be [Logged on](#) to comment or reply to a post.

---



**Krishna Kishor Kammaje**

December 14, 2013 at 2:43 am

Hey [Masoud](#),

Very useful feature. Congrats to your team for building this into ABAP stack. I am excited by its possibilities.

I am building SAP UI5 applications based on NW Gateway services. One of the very required feature is pessimistic resource locking. Example: If I am editing a purchase order in UI5 app, no other user using UI5 app should be able to change it.

This is my thought process after reading your blog.

Within my UI5 application, as soon as someone enters into change mode, I would create a websocket connection for that client ([UI5 library provides an API](#)) and send the resource id (say PO number). In the server side I would make an entry in a Z table (with a generated GUID connection ID and PO number) and give back a success message to the client. If the Z table already has an entry for that PO, then the server would refuse the lock and push a suitable message. Whenever the user closes the browser/tab, I suppose that the websocket connection would end, and in the on\_close method, I would remove the entry from the Z table. If the user is inactive for more than 10 mins, I would end the websocket connection in the client javascript.

As I understand Z Table would not be required in a stateful websocket, since that should allow me to make use of SAP Lock objects.

How does this sound?

Please let me know your comments about this approach.

Thanks

Krishna

Like (0)



Former Member

April 23, 2017 at 5:07 pm

Hi Krishna,

We too have similar requirements for Pessimistic Locking. Please share your insights of how you were able to achieve this solution. Also find below some data points for your consideration.

- All the objects are Z-Objects so no locking is expected from anywhere else, apart from UI5 browser.
- APC/AMC supported in our current package
- What would happen if user simply closes his system or there is a timeout? Can web sockets address these cases?

I was planning to use your above suggested approach for locking.

Regards,

Samson

Like (0)



Masoud Aghadavoodi Jolfaei | Post author

February 20, 2014 at 7:30 pm

Hi Krishna,

your proposal sounds good as far as those lock entities, e.g. purchases orders, are not accessed by other classical transactions, e.g. from within SAPGUI or RFC, using ABAP enqueue based locks. In that case the enqueue locks and your custom table for recording of locks are not anymore in sync and this could lead to side effects, e.g. accessing the order though the order is already marked as locked in the Z\* table .

Best regards,

Masoud

Like (0)



Krishna Kishor Kammaje

February 22, 2014 at 9:26 am

Thanks Masoud for your insight.

Like (0)



Former Member

May 14, 2015 at 10:53 am

Hi Masoud,

Thanks for this blog.

I have also implement same web socket in gateway service. When I am calling from browser it's calling Accept method then immediately it's calling close method.

how I will handle close method . it should call on only when user explicitly call close or browser close

Thank.

Like (0)



Former Member

March 1, 2014 at 3:22 pm

Hi Masoud,

I was trying your example. The part of send/receive works perfectly. But now I wanted to experiment further based on what you have written about **"exchange of messages between either user sessions"**. I first tried to run two instances of same application and in parallel browser windows and was trying to send message hoping it should also be received in the other window. But this does not work. I also ceated a simple html page using the websocket URL just created, but this also works in its own context it only listens and transmits its own message.

Would the class need to be enhanced further. I tried this on Basis 740 SP-Level 0000.



Warm Regards

Shadab Shafiq

Like (0)



Masoud Aghadavoodi Jolfaei | Post author

March 2, 2014 at 5:16 pm

Hi Shadab,

this blog is one of the up-coming 3-4 articles around ABAP Channel capabilities. As I mentioned in the introduction:

...

*The ABAP channels supports the following use-cases:*

1. *ABAP Push Channel (APC): The WebSocket integration in ABAP.*
2. *ABAP Messaging Channel (AMC): Eventing framework for messages exchange between different ABAP sessions based on publish/subscribe channels.*
3. *Collaboration scenario: Using APC and AMC to push messages from ABAP sessions to WebSocket clients by binding publish/subscribe channels to the WebSocket connection.*

*In this article we focus on the WebSocket framework in ABAP called ABAP Push Channel.*

...

Well I am writing at the moment a blog about ABAP Messaging Channel (AMC) and additionally I will spend time for the collaboration article which will contain the information you are looking for. But in the mean time you can have a look into one of my example in the system. Just execute the APC test application "http(s)://<host>:<port>/sap/bc/apc\_test/ping" in Chrome/Firefox browser, preferably in several browser windows. Before executing the URL you have to activate the paths "/sap/bc/apc\_test/ping" and "/sap/bc/apc/sap/ping" in the transaction SICF. The proposed URL executes the APC application "PING" (look into the html source code) which itself take use of AMC application APC\_SIMPLE\_TEST to exchange of messages between different WebSocket connections. You may debug the ON\_START and ON\_MESSAGE methods of the APC application "PING" during execution of the URL.

Best regards,

Masoud

Like (0)



Former Member

March 3, 2014 at 8:05 am

Hi Masoud,

Thanks.

Please update this article with followup link to your new articles. Really looking forward to it.

Warm Regards

Shadab

Like (0)



Vikas Lamba

June 8, 2014 at 2:37 pm

Hi Masoud,

Excellent blog. Thanks for sharing.

I was trying what you did and have a question now. I see that all the test client examples are using a Web Dynpro application which is of course hosted in the same NW server and have no security issue communicating.

I tried to write a JS page to make a client and then realized that JS WS API does not provide any way to pass authorization information to server. So I was getting HTTP 401 instead of a WS upgrade.

Can you share some light on how can I use a JS client to open a WS connection with ABAP and does it need any additional configurations on server to enable this. I Have NW 7.4 SP06 installed.

Cheers,

Vikas

Like (0)



Masoud Aghadavoodi Jolfaei | Post author

June 8, 2014 at 3:36 pm

Hi Vikas,

you can pass the logon data (just for testing) as cgi form parameters, i.e. sap-client=<client number>&sap-user=<username>&sap-password=<password>. If the JS WS client page does not reside on the same ABAP system, then you have to maintain the cross origin access white list accordingly (refer to cross-origin-access

in APC security section). Each WS access rejection due to cross origin access is reported in ABAP log, i.e.transaction SM21 on the respective application server.

Alternatively you can create the JS WS client in ABAP system and this either as a BSP application or as a free style HTTP handler (just have look into the SICF handler maintained for the path /sap/bc/apc\_test/ping (also have a look to my reply to Shadab)).

Best regards,

Masoud

Like (0)



Horst Keller

July 24, 2014 at 4:21 pm

*Can you share some light on how can I use a JS client to open a WS connection with ABAP*

In my blog [ABAP News for Release 7.40 – WebSocket Communication with ABAP Channels](#) I show an example, where I drilled down the web client that communicates with ABAP to the absolute minimum – no BSP, no WDY, only a slim JS. Maybe that's what your looking for or is it already too primitive? B.T.W. if you are on NW 7.4 SP06, you find that example ready for execution as program DEMO\_APC in the ABAP Example Library (see [APC-Example](#)).

Best

Horst

Like (0)



Vikas Lamba

August 3, 2014 at 10:42 am

Hi Horst,

Thanks for the example. I was already able to get this working.

But just to describe my goal I was trying to use WS connections to achieve a real time push to browser based applications on mobile phones (iOS).

I hit a dead end when I realized that iOS terminates all WS connections originating from a mobile device when the app goes to background.

Is there any solution that you are aware of for this problem?

Regards,

Vikas

Like (0)



Masoud Aghadavoodi Jolfaei | Post author

August 3, 2014 at 1:50 pm

Hi Vikas,

the only workaround I am aware of is that you try to establish the WebSocket connection in WebSocket onclose event again (in order to overcome an endless loop just limit the number of trials). This event is processed as soon as the application gets activated again. This will help to reload the content through WebSocket connection.

A real event-driven push can be achieved with the push platforms, e.g. SAP/Sybase Unwired Platform (SUP) which also take use of apple push notification service in case of iOS to notify/awake the application.

Best regards,

Masoud

Like (0)



Former Member

March 6, 2017 at 8:13 am

Hi, Vikas

We also run into the similar issue as you.

We are making some web application deployed in public internet. And it is confused that how to let the ERP system in internal network to authenticate the connection from outside.

Do you use the wss URL like

`wss://ldai3ebj.wdf.sap.corp:44315/sap/bc/apc/sap/zhar_apc_center?sap-client=<client number>&sap-user=<username>&sap-password=<password>?` Do you encode the password in BASE64?

And do you make other configuration in ERP system except the cross-domain config?

Like (0)



**Michael Stürmer**

June 24, 2014 at 9:51 am

Hi Masoud,

while experimenting, I wanted to do some connection-specific cleanup in the ON\_CLOSE-method and ended up enhancing CL\_APC\_MANAGER=>DISPATCH to get to the initial request. Is there any official way to access the initial request from within ON\_CLOSE?

Also, I noticed some WebSocket-related (but unfortunately not released) notes in the web dispatcher patch documentation. Do you know if/when there'll be official support for WebSockets in web dispatcher?

Thanks,

Michael

Like (0)



**Masoud Aghadavoodi Jolfaei** | Post author

June 26, 2014 at 7:35 am

Hi Michael,

ON\_CLOSE/ON\_ERROR handling of APC has been enhanced with NW 740 SP08 release. In the 740 SP08 release contains ON\_CLOSE/ON\_ERROR method the initial request interface parameter as well. The solution to establish WebSocket context information for a WebSocket connection is to identify the WebSocket connection id, which is stable for the whole life time of the connection. The connection is then used as key to temporarily persist the context information on the backend, either by using a (buffered) database table or shared objects areas. Usually you would establish the context information for the WebSocket connection, i.e. APC application context, either in ON\_START or in ON\_MESSAGE method. And the cleanup should take place in ON\_CLOSE method.

By the way we do not guarantee that the ON\_CLOSE method will be executed under all circumstances, as there are worst-case scenarios which this hinder us to execute any ABAP coding.

For the identification of the WebSocket connection id following steps has to be applied:

1. For NW 740 SP05 release (for correct extended passport (EPP) handling for WebSocket) please apply the kernel patch level 35, i.e. download from [service.sap.com/swdc](http://service.sap.com/swdc) => Browse our Download Catalog => Additional Components => SAP Kernel
2. Use following ABAP coding in ON\_START/ON\_MESSAGE or ON\_CLOSE methods to retrieve the connection id out of EPP:

```
DATA: lv_connection_id TYPE epp_root_context_id.
```

```
TRY.
```

```
DATA(lo_epp_global) = cl_epp_global_factory=>get_section( ).
```

```
lv_connection_id      = lo_epp_global->get_root_context_id_as_uuid( ).
```

```
CATCH cx_epp_error INTO DATA(lx_epp_error).
```

```
    " bad luck/should not happen !!!
```

```
ENDTRY.
```

Web Dispatcher supports WebSocket out of the box without any additional configuration since 7.40 patch level 15 and with 7.41 patch level 0.

Enjoy the ABAP Channels,

Masoud

Like (0)



Daniel Rothmund

August 8, 2014 at 8:24 am

Hi ,

what are the correct settings to use the APC with an SAP WebDispatcher infront ?

I doesn't find anything about it at [help.sap.com](http://help.sap.com)

Regards

Like (0)



Masoud Aghadavoodi Jolfaei | Post author

August 8, 2014 at 9:27 am

Hi Daniel,

Web Dispatcher supports WebSocket out of the box without any additional configuration since 7.40 patch level 15 and with 7.41 patch level 0.

Kind regards,

Masoud

Like (0)



Daniel Rothmund

August 8, 2014 at 10:42 am

thx now it works

Like (0)



Former Member

September 11, 2014 at 6:26 pm

Hi Masoud,

First of all thank you for writing this blog! I find it very useful as well as the technology it relates about.

Now to get to the point of this question, I would like to use APC in some applications, but I have a few concerns regarding limitations in terms concurrent connections to an APC. Is there any hard-coded restriction in terms of the number of concurrent connections to an APC?

Best regards,

Sergiu

Like (0)



Masoud Aghadavoodi Jolfaei | Post author

September 12, 2014 at 10:07 am

Hi Sergiu,

the limitation based on the application server/ICM/WebDispatcher profile parameter configurations. In our lab tests with the respective server we reached 20000 parallel WebSocket connections, which in certain cases led to memory issues in backend. The respective profile parameter for ICM/WebDispatcher are:

rdisp/max\_websocket\_connections=20000

icm/max\_conn=20000

Best regards,

Masoud

Like (0)



Former Member

[September 26, 2014 at 4:28 am](#)

Can ABAP Push Channel framework be configured in SAP Netweaver 7.30 ?

Like (0)



[Masoud Aghadavoodi Jolfaei](#) | Post author

[September 26, 2014 at 7:09 am](#)

Hi,

this feature is only available in AS ABAP 740 releases.

Best regards,

Masoud

Like (0)



Former Member

[October 24, 2014 at 3:27 pm](#)

Hi Masoud,

Really nice Blogs. I really appreciate the time you have put in to show make it available for SDn users. I was experimeting a bit around this to get rid of the notification service used in my WD application. So right now I have a WD application with a HTMLISLAND and have a JavaScript added into it as a Mime Object. I am



creating a Websocket object within my JS function. What I am trying to do is to pass a value from my actual WD application to the JS function and from JS function I am trying to pass on the value to the APC~ON\_START method. The first part seems to be easy enough. But I am finding it difficult to pass any value to the ON\_START method. I did checked the Websocket API but I didnt find anything useful for my purpose.

I would really appreciate if you could put some inputs here.

Regards,

Manish

Like (0)



Masoud Aghadavoodi Jolfaei | Post author

October 24, 2014 at 4:10 pm

Hi Manish,

you can pass any name-value pairs as part of the WebSocket URL (and query string) to the APC application, such as `wss://host:port/sap/bc/apc/sap/<myapplication>?name1=value1&name2=value2`. In the APC application `<myapplication>` in ON\_START or ON\_MESSAGE the parameters can be accessed via context interface `IF_APC_WS_CONTEXT` and as follow:

METHOD `if_apc_ws_extension~on_start`.

DATA: `lo_binding` TYPE REF TO `if_apc_ws_binding_manager`.

DATA: `lo_request` TYPE REF TO `if_apc_ws_initial_request`.

DATA: `lx_apc_error` TYPE REF TO `cx_apc_error`.

DATA: `lt_fields` TYPE `tihttpnvp`.

DATA: `lv_value` TYPE `string`.

DATA:

TRY.

`lo_binding = i_context->get_binding_manager( ).`

`lo_request = i_context->get_initial_request( ).`

“determine cgi parameters aka form fields

```
lo_request->get_form_fields( CHANGING c_fields = lt_fields ).
```

```
lv_value = lo_request->get_form_field( 'name1' ).
```

```
CATCH cx_apc_error INTO lx_apc_error.
```

```
data(lv_message) = lx_apc_error->get_text( ).
```

```
MESSAGE lv_message TYPE 'E'.
```

```
ENDTRY.
```

```
ENDMETHOD.
```

Just check out in your development system the ON\_START & ON:MESSAGE methods of the APC application ECHO in the SAPC transaction as example

Cheers,

Masoud

Like (0)



Klaus Keller

October 29, 2014 at 2:18 pm

Hi Manish,

To pass values from Web Dynpro to the HTMLIsland (JavaScript), you'll need to add a script call. See the HTMLIsland Developer Guideline for examples on how to do this: <http://scn.sap.com/docs/DOC-34098>

Regards,

Klaus

Like (0)



Former Member

[December 5, 2014 at 12:59 pm](#)

Hello,

I have created mu APC just as described in this document.

However when I want to 'Test' my APC from Tx SE80, I get the error 'Function not yet implemented'

Any ideas what can be the problem ?

Regards,

Erwin

Like (0)



[Masoud Aghadavoodi Jolfaei](#) | Post author

[December 8, 2014 at 8:44 am](#)

Hi Erwin,

we couldn't reproduce the issue in different ABAP Application Server support packages (SP) >= SP5. If you still observe the issue and not able to identify the reason please create a CSN ticket for the component BC-MID-AC and provide the affected system release (ABAP Application Server support package ??).

Best regards,

Masoud

Like (0)



Former Member

[December 8, 2014 at 9:21 am](#)

Hello,

Our system is currently SP2

Do we need SP5 before we can use the Abap Push Channel ?

Because I was able to create the pushchannel fine.

He generated the class and service.

The service I needed to activate manually in SICF

Best regards,

Erwin

Like (0)



[Masoud Aghadavoodi Jolfaei](#) | Post author

[December 8, 2014 at 10:23 am](#)

Hi Erwin,

we have released the “basic functionality” for the ABAP Channels with SP5. I personally recommend – if/as soon as possible – to use at least SP8. As with the SP8 we have established additional capabilities for the ABAP Push Channel (see the CSN note 2105389).

Best regards,

Masoud

Like (0)



Former Member

[January 30, 2015 at 9:35 am](#)

Thank you. this really helped..

Like (0)



Former Member

[May 14, 2015 at 6:18 am](#)

Hi [Masoud Aghadavoodi Jolfaei](#),

Thanks for Sharing The Knowledge. and i have one Doubt in my Development.

is it Possible To Pass any parameter from Back end and Retrieve it in WD Componet??

If it so, how to Achieve it??

Pls Help me.

Thanks in Advance.

Like (0)



Masoud Aghadavoodi Jolfaei | Post author

May 16, 2015 at 1:31 pm

Hi Siva anand,

I do not understand the (end to end) use case. If you like to push messages/parameters from an ABAP session to another one (WD component), wheres the target session is a stateful session (here Web Dynpro session), in this case the [ABAP Messaging Channels](#) provide the necessary infrastructure. Alternatively using the [ABAP Channels collaboration scenario](#) you can push messages/parameters from an ABAP session to your Web Dynpro UI (HTML island) running in browser and additionally forward it to your Web Dynpro session/component.

I also recommend to you to place the Web Dynpro related subjects in the Web Dynpro discussion room.

Best regards,

Masoud

Like (0)



Former Member

May 16, 2015 at 6:36 am

Hi Masoud,

Thanks for the wonderful blog series.

I built a UI5 application using websockets and I integrated this application into our SAP Enterprise Portal. When the UI5 application is executed from the portal the websocket connection with the backend was getting established successfully. We tried this from multiple PCs and it worked.

Next we modified the application, redeployed it in the backend and re-run the application from the portal and am facing the following issues:

a. The websocket connection is NOT getting established and I am getting the error '**Error during websocket handshake: unexpected response code: 403**'.

b. Then i tried executing the application from another PC and in this case the websocket connection is getting established however the modifications made to the application are not getting reflected.

Then I tried executing the UI5 application directly from the browser and in this case it worked correctly.

Would appreciate you help in trying to resolve the issues I am getting when executing the UI5 application from the portal.

Server versions: Portal: SAP EP 7.4, Backend ECC – ECC6 EHP7 SP06

regards

Nitesh

Like (0)



Masoud Aghadavoodi Jolfaei | Post author

May 16, 2015 at 1:44 pm

Hi Nitesh,

the response code 403 is usually processed because of a cross origin access issue (see also the section APC Security of my blog). You will find in the system log of the corresponding ABAP application server, i.e. in the transaction SM21, further information to the APC issues. I also recommend to read the SAP note "[2123214 – Error handling for ABAP Channels using SYSLOG entries](#)" any apply the necessary cross-origin configuration in the affected system.

Best regards,

Masoud

Like (0)



Former Member

March 3, 2016 at 2:24 pm

Hi Masoud,

Thanks for your contribution. It really helps. Right now I'm developing an UI5 application with ABAP push channel and the APC works fine when test with WebDynpro WDR\_TEST\_APC. But I'm stuck when I try to integrate it with UI5. Here is my JS code

```
/**
 * Init the configuration of ABAP Push Channel
 */

//check if websocket supported
if(!sap.ui.Device.support.websocket){
    sap.m.MessageBox.show(
        "Sorry, Your SAPUI5 version doesn't support WebSocket.", {
            icon: sap.m.MessageBox.Icon.ERROR,
            title: "WebSocket not supported",
            actions: [sap.m.MessageBox.Action.OK]
        });
    return;
}

//establish websocket connection
this.oWs = new SapPcpWebSocket("/sap/bc/apc/sap/zqchart_ws", SapPcpWebSocket.SUPPORTED_PROTOCOLS.v10);

//register callback functions on WebSocket Events
this.oWs.attachOpen(function(e){
    console.log("connection created");
});

this.oWs.attachMessage(function(oEvent){
    //Message from server is coming in
    console.log(oEvent);
});
```

And I configured the proxy in neo-app.json so that it can redirect request.

```
{

    "path": "/sap/bc/apc",

    "target": {

        "type": "destination",

        "name": "ER9CLNT001",

        "entryPath": "/sap/bc/apc"

    },
```

```
"description": "ER9 client 001"
```

```
}
```

But the app ends with an error when running in Chrome.

**WebSocket connection to 'wss://webidetesting3973187-fiori.dispatcher.neo.ondemand.com/sap/bc/apc/sap/zqchart\_ws' failed: HTTP Authentication failed; no valid credentials available.**

And the error changed to **Firefox can't establish a connection to the server at wss://webidetesting3973187-fiori.dispatcher.neo.ondemand.com/sap/bc/apc/sap/zqchart\_ws**. when running in Firefox. Do you have any idea about this issue?

Best Regards,

Kevin Zhang

Like (0)



**Masoud Aghadavoodi Jolfaei** | Post author

March 15, 2016 at 11:59 am

Hi Kevin,

I assume you are using WebIDE to develop your UI5/Fiori application in HCP. If so up to now the cloud connector and Helium and its destination service (in HCP) does not support any WebSocket communication. Here you have to setup the connection directly to the ABAP backend system.

Cheers,

Masoud

Like (0)



**Kenneth Moore**

March 3, 2016 at 7:12 pm

Hello Masoud,



When I test the Push Channel it does not connect. I push the connect button, but nothing happens and there is no message. Any ideas?

Thanks,

Kenneth

Like (0)



[Masoud Aghadavoodi Jolfaei](#) | Post author

[March 15, 2016 at 12:01 pm](#)

Hi Kenneth,

in Chrome browser you usually find in the developer tools (F12) -> Network -> WS log entries regarding the connection setup issues.

Cheers,

Masoud

Like (0)



[Kenneth Moore](#)

[March 15, 2016 at 12:35 pm](#)

Hi Masoud,

There are no log entries.

Like (0)



[Masoud Aghadavoodi Jolfaei](#) | Post author

[March 22, 2016 at 7:54 am](#)

Hi Kenneth,

this is very strange. This means somehow that the browser has problems to process the WS(S) command. Here you have to analyze/debug the JavaScript page in Chrome.

Cheers,

Masoud

Like (0)



Daniel Rothmund

May 17, 2016 at 3:12 pm

Hello Masoud,

ist it possible to have fixed user for a wss connection ?

I tried to add a service user to sicf but I get an 403 from icm.

Regards

Like (0)



Masoud Aghadavoodi Jolfaei | Post author

May 17, 2016 at 4:00 pm

Hi Daniel,

sure, in that case you have to choose in the respective (external) SICF service and in the tab "Logon Data" as "Procedure" the option "Required with Logon Data". By the way the response code 403 arises (I assume) since the cross origin policy check of APC failes. Here you have to apply the SAP note "[2123214 – Error handling for ABAP Channels using SYSLOG entries](#)".

Cheers

Masoud

Like (0)



Daniel Rothmund

May 18, 2016 at 5:15 am

Thx Masoud ,

the problem was the cross origin policy !

And also thanks for the hint with the note. I did not know "RS\_AC\_READ\_SYSLOG".

BR

Daniel

Like (0)



Paul Hardy

June 24, 2016 at 7:35 am

Dear All,

In the above comments I have seen several people attempting to add code to a UI5 application in order to initiate a web socket connection and then send a message from a UI5 application to SAP.

As this works with the test tool in transaction SAPC it must be possible, but no-one here at least seems to be able to get this to work.

Has anyone in fact managed to write some code in any part of a UI5 application – be it XML or JS – that manages to establish a connection and then send a message from the UI5 front end to the SAP back end?

Cheersy Cheers

Paul

Like (0)



Masoud Aghadavoodi Jolfaei | Post author

June 24, 2016 at 8:43 am

Hi Paul,

I have recently released a new blog called [ABAP Channels Examples](#). One of my intention for this blog was to provide several “simple use cases” for the ABAP Channels developer, including the involved ABAP and UI coding in the system. So I recommend to you to look at the **Sohbat Slider** example which from UI perspective consists of a single BSP/Fiori (`/sap/bc/bsp/sap/sohbat_slider`) application.

Cheers,

Masoud

Like (0)

---



Dren Selimi

October 10, 2016 at 1:41 pm

Hello Masoud,

I am having trouble receiving a message from javascript in the method ON\_MESSAGE, for example, using the code on javascript that you provided above I can connect to the WebSocket and I can see that if I use a break-point in the ON\_START method in SAP, but the part for " ws.send("hello);" " won't trigger ON\_MESSAGE, is that what should happen, or I have got it wrong, I that is what should happen why doesn't happen, where do you think is my problem ?

Thanks, Dren

Like (0)



Masoud Aghadavoodi Jolfaei | Post author

March 14, 2017 at 10:22 am

Hi Dren,

sorry for the late response. Please have a look into the [FAQ – ABAP Channels](#).and also note [2353453](#).

Cheers,

Masoud

Like (0)

---



Former Member

July 26, 2017 at 5:50 am

Hi Masoud,

I'm getting a 501 (Not Implemented) error when trying to connect to the WS. What could be causing this, and how could I go about debugging?

Thanks,  
Kyle

Like (0)



Masoud Aghadavoodi Jolfaei | Post author

January 6, 2018 at 10:47 am

Hi Kyle,

sorry for late response. I assume there is something wrong with the creation of you application. Just refer to the [FAQ – ABAP Channels](#), and the related question “My APC WebSocket connection to the ABAP application server does not work. Are there known pitfalls? What are the first steps for troubleshooting?”.

Good luck!

Masoud

Like (0)



Serdar Simsekler

August 1, 2017 at 9:14 am

Hello Masoud

Thanks for the blog series. Can I ask how we can have the ABAP side send messages directly to the e.g. UI5 application through the defined APC without AMC?

So, a running ABAP session decides to send a message to any/all frontend applications hooked on the defined APC, i.e. established connection to the APC service, and just sends it without an incoming message from the frontend applications.

Thanks

Serdar

Like (0)



Masoud Aghadavoodi Jolfaei | Post author

January 9, 2018 at 1:58 pm

Hi Serdar,

sorry for the late response. From ABAP Server release  $\geq 7.50$  we have established the concept of system-wide access or attach handle to be able to push data from a session to an existing WebSocket connection. For this please refer to teh documentation and examples available [here](#).

Cheers,

Masoud

Like (0)



Former Member

January 21, 2018 at 11:44 am

Hello Masoud,

If I call a BAPI inside the method on\_message and that BAPI internally calls a function module which raises an error message then the session ends abruptly. In normal report programs or classes, that is not the case and standalone BAPI is also executed successfully.

Is it normal behavior in case of APC?

Thanks,

Manish

Like (0)



Masoud Aghadavoodi Jolfaei | Post author

January 21, 2018 at 12:39 pm

Hi Manish,

if you are using an APC stateless application, then this behavior should not happen, abstractly speaking. It could be that the BAPI runtime checks some session type specific context information which are not available in an APC session. That is, if you run a BAPI from within a SAPGUI and thus SAPGUI session in SE38 or SE24 in this case the SAPGUI session has GUI (interaction) capability which is not given in an APC session. What is the error text of the raised message ? I would propose to debug both situation, i.e. good and bad case and check what causes the error.

Cheers,

Masoud

Like (0)



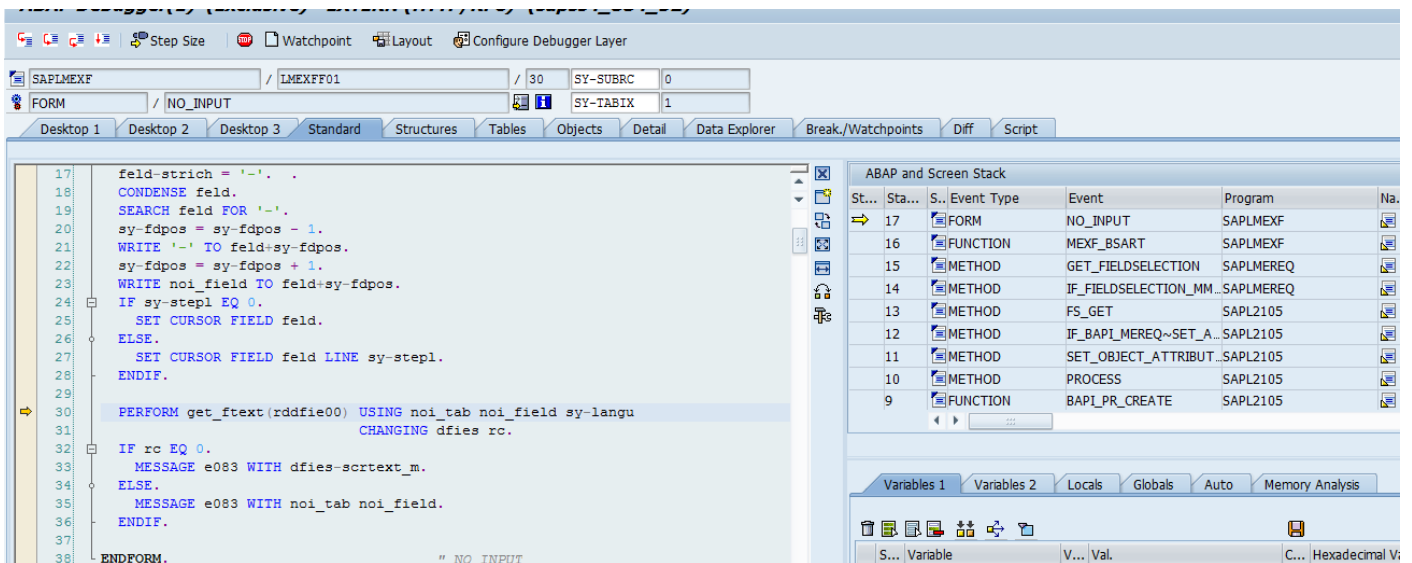
Former Member

January 21, 2018 at 1:57 pm

Hello Masoud,

Thanks for the reply.

I am trying to use 'BAPI\_PR\_CREATE' to create purchase requisition. I suspect that it is the same as you suggested.



In the above screenshot, it raises an error at line 33 and session ends here itself abruptly.

When I call the same BAPI via a report program, the error is still raised but program continues.

Do you have any idea how this can be avoided?

Thanks,

Manish

Like (0)



Masoud Aghadavoodi Jolfaei | Post author

January 21, 2018 at 2:15 pm

Hi Manish,

the E/A/X message handling in ABAP is context dependent, i.e. E message handling in SAPGUI ia SE38, is different then in a Batch Input or an RFC or APC session. For a detail analysis of the reported issue you have to open/create a customer message@SAP for the responsible component, e.g. purchase requisition. A workaround, which might help, without knowing the end to end scenario, is the execution of the BAPI not locally in APC session but as a synchronous RFC out of APC session via CALL FUNCTION 'BAPI\_PR\_CREATE' DESTINATION 'NONE'.

Good luck !

Masoud

Like (0)



Former Member

January 22, 2018 at 8:43 am

Dear Masoud,

Thanks for your help. I tried with the suggested approach and it worked like a charm. I'll raise an SAP note for the message part in APC.

Thanks again for your quick inputs.

Regards,

Manish

Like (0)

## Share & Follow

[Privacy](#)

[Terms of Use](#)

[Legal Disclosure](#)

[Copyright](#)

[Trademark](#)

[Cookie Preferences](#)

[Sitemap](#)

[Newsletter](#)