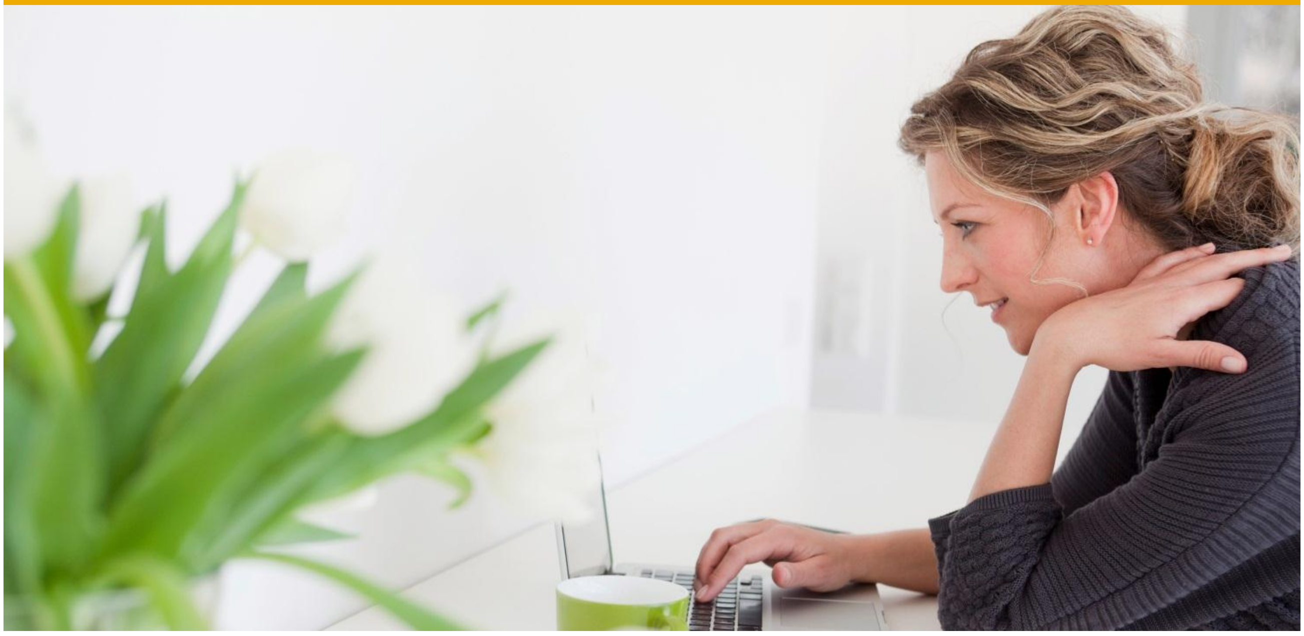


Persistence Layer and Runtime of the SAP Application Interface Framework

TABLE OF CONTENTS

With the SAP Application Interface Framework 2.0 an own runtime and persistence layer is delivered. This enables you to process and monitor data independent of any technology. For example you can trigger processing in AIF with a report or a function module. Additionally, the data will be persisted on the SAP Application Interface Framework's own persistence layer. Therefore, the data content will be visible in *Monitoring and Error Handling*. Furthermore, it will be possible to restart the data message in case an error occurred during processing.

In order to processes the data with the SAP Application Interface Framework's runtime and persistence layer, you have to at least perform the following steps:

- Create source structure and destination structure
- Define an interface in the SAP Application Interface Framework
- Maintain XML Application Engine and XML Persistence Engine
- Define a structure mapping for your interface
- Trigger processing with the SAP Application Interface Framework's runtime by calling:
    - /AIF/CL_ENABLER_XML=>TRANSFER_TO_AIF to trigger processing of a single message
    - /AIF/CL_ENABLER_XML=>TRANSFER_TO_AIF_MULT to trigger processing of multiple messages

PROCESS DATA WITH THE SAP APPLICATION INTERFACE FRAMEWORK'S RUNTIME

Scenario Overview

In this chapter an interface example will be developed that will change a customer's data by calling BAPI_FLCUST_CHANGE. The processing of the data will be triggered by a report. For reasons of simplicity the report will only consist of some input fields and will directly pass the data from input screen to the SAP Application Interface Framework. The SAP Application Interface Framework will perform some mappings, since the data passed by the report does not directly fit the data of BAPI_FLCUST_CHANGE. If the data could be mapped successfully the action that will call BAPI_FLCUST_CHANGE is executed. After the processing was triggered by the report you should be able to see the messages in *Monitoring and Error Handling*. You can display the data processed by the report and perform error handling.

Dictionary Objects and Report for Message Processing

First of all you should create the raw data structure. The structure should include the customer number. However, how the rest of the structure looks like depends on the data you would like to change. For example the structure could look as follows:

Component	Component Type
PERSONAL_DATA	ZAIF_X102_PERS_DATA
ADDRESS_DATA	ZAIF_X102_ADDRESS_DATA
COMMUNICATION_DATA	ZAIF_X102_COMM_DATA

Sub-Structure PERSONAL_DATA (ZAIF_X102_PERS_DATA):

Component	Component Type
CUSTOMERNUMBER	S_CUSTOMER
NAME	S_CUSTNAME
FORM	S_FORM
BIRTHDATE	S_BIRTHDAT
CUST_TYPE	S_CUSTTYPE
DISCOUNT	S_DISCOUNT

Sub-Structure ADDRESS_DATA (ZAIF_X102_ADDRESS_DATA):

Component	Component Type
STREET	S_STREET
POBOX	S_POSTBOX
POSTCODE	POSTCODE
CITY	CITY
COUNTR	S_COUNTR
COUNTR_ISO	S_COUN_ISO
REGION	S_REGION

Sub-Structure COMMUNICATION_DATA (ZAIF_X102_COMM_DATA)

Component	Component Type
PHONE	S_PHONENO
EMAIL	S_EMAIL
LANGU	S_LANGU2
LANGU_ISO	S_LANG_ISO

Now you can create the report to trigger the message processing. The report should have the customer number and the data of the customer that should be changed as import parameters. Furthermore, a variable that has the type of your raw structure is needed. Move the input data to this variable. In order to trigger the processing of the data in the SAP Application Interface Framework you have to call static method TRANSFER_TO_AIF of class /AIF/CL_ENABLER_XML. The method has only one mandatory input parameter IS_ANY_STRUCTURE. All other parameters are optional. Pass the structure with the format of your raw data structure to IS_ANY_STRUCTURE. The report could look as follows:

```
REPORT  zaif_test_x102_change_flcust.

*customer and data that will be changed during message processing
PARAMETERS: p_custnr TYPE scustom-id,
            p_name   TYPE s_custname LOWER CASE,
            p_street TYPE s_street,
            p_pcode  TYPE postcode,
            p_city   TYPE city,
            p_countr TYPE s_countr,
            p_phone  TYPE s_phoneno,
            p_email  TYPE s_email.


*ls_input should have the type of raw data structure
DATA: ls_input TYPE zaif_x102_raw_flcust_change.

*move input data to raw data structure
ls_input-personal_data-customernumber = p_custnr.
ls_input-personal_data-name            = p_name.

ls_input-address_data-street = p_street.
ls_input-address_data-city   = p_city.
ls_input-address_data-postcode = p_pcode.

ls_input-communication_data-phone = p_phone.
ls_input-communication_data-email = p_email.

*pass data to AIF enabler for XML to trigger message processing
/aif/cl_enabler_xml=>transfer_to_aif(
  EXPORTING
    is_any_structure = ls_input ).
```

Create an SAP structure for your interface. The SAP Structure should fit to the importing parameters of BAPI_FLCUST_CREATE. Your SAP Structure could look like in the table below:

Component	Component Type
CUSTOMERNUMBER	S_CUSTOMER
CUSTOMER_DATA	BAPISCUNEW
CUSTOMER_DATA_X	BAPISCUNEX

Interface Customizing

Now you have to define your interface in customizing activity *Define Interfaces*. To access the customizing of the SAP Application Interface Framework call transaction /AIF/CUST.

Insert an interface name and a version. Insert your *SAP Data Structure* and your *Raw Data Structure*. Save the interface.

Furthermore, you have to maintain the correct engines to handle your messages in *Monitoring and Error Handling*. Select following engines in *Additional Interface Properties→Specify Interface Engines*:

- Application Engine: XML
- Persistence Engine: XML
- Selection Engine: AIF Index Tables
- Logging Engine: AIF Application Log

To create a structure mapping go to customizing activity *Define Structure Mappings* and select your new interface. Create structure and field mappings in order to move your data from the raw data structure to the SAP data structure.

Create a structure mapping from source root structure to the destination root structure. Note that *Source Structure* and *Destination Structure* field will only contain a space in case of a root structure mapping.

Create field mappings for your structure mapping in *Define Field Mappings*.

Field in Destination Structure	Fieldname 1
CUSTOMERNUMBER	PERSONAL_DATA-CUSTOMERNUMBER
CUSTOMER_DATA-CITY	ADDRESS_DATA-CITY
CUSTOMER_DATA-COUNTR	ADDRESS_DATA-COUNTR
CUSTOMER_DATA-CUSTNAME	PERSONAL_DATA-NAME
CUSTOMER_DATA-EMAIL	COMMUNICATION_DATA-EMAIL
CUSTOMER_DATA-PHONE	COMMUNICATION_DATA-PHONE
CUSTOMER_DATA-STREET	ADDRESS_DATA-STREET
CUSTOMER_DATA-POSTCODE	ADDRESS_DATA-POSTCODE

Since the BAPI will only change data if the corresponding field is set to ‘X’ in input parameter CUSTOMER_DATA_X, you have to make sure that the corresponding field in SAP structure CUSTOMER_DATA_X is set to ‘X’. The easiest way to achieve this is to define fix values and set the flag for every field that should be changed.

Field Name	Value
CUSTOMER_DATA_X-CITY	X
CUSTOMER_DATA_X-COUNTR	X
CUSTOMER_DATA_X-CUSTNAME	X
CUSTOMER_DATA_X-EMAIL	X
CUSTOMER_DATA_X-PHONE	X
CUSTOMER_DATA_X-POSTCODE	X
CUSTOMER_DATA_X-STREET	X

Note: Using the fix value will always change the content of the corresponding field, even if the corresponding field is empty. A more dynamic approach is to create field mappings with conditions for the fields of CUSTOMER_DATA_X. The condition will set the value to X if the corresponding field of CUSTOMER_DATA is not empty.

Now you have to assign an action to your interface. Click *Assign Action* and enter a namespace and an action name. If the action does not exist, you will be asked if you want to create it. Double clicking the action’s name will forward you to the customizing activity *Define Actions*. Select *Commit Mode COMMIT WORK* and *Commit Level After Each Function*. Assign an action Function in *Define Function*. Enter a function module if the function module does not exist it will be created. Implement the function module to call BAPI_FLCUST_CHANGE. You can change the type of changing parameter DATA to fit the type of your SAP structure. Pass the data of your SAP structure to the function module.

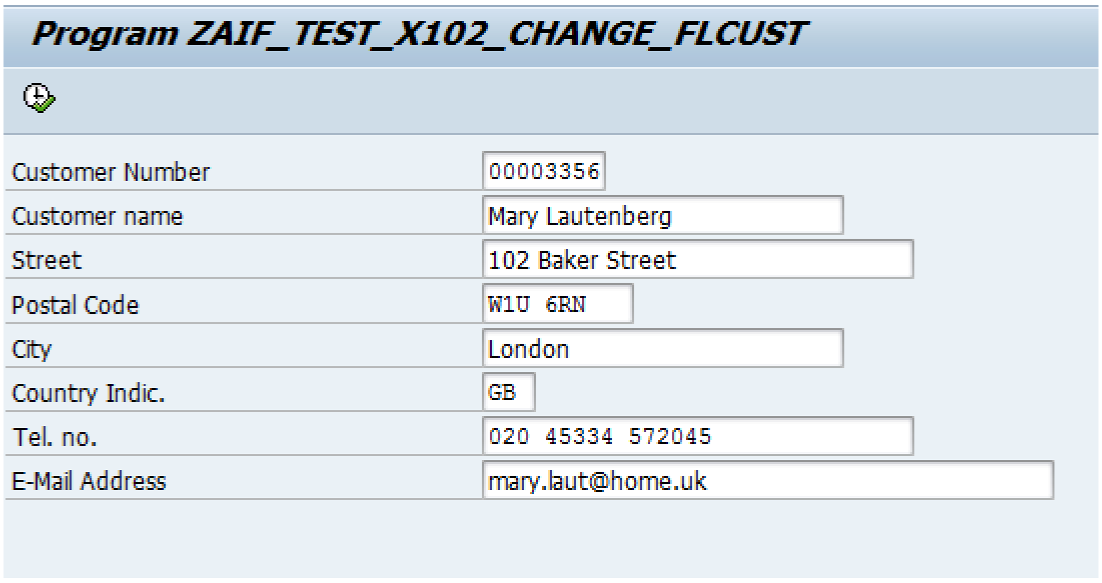
```
FUNCTION z_flcustomer_change .
*"----------------------------------------------------------------------
*"*"Local Interface:
*"  IMPORTING
*"     REFERENCE(TESTRUN) TYPE  C
*"     REFERENCE(SENDING_SYSTEM) TYPE  /AIF/AIF_BUSINESS_SYSTEM_KEY
*"       OPTIONAL
*"  TABLES
*"      RETURN_TAB STRUCTURE  BAPIRET2
*"  CHANGING
*"     REFERENCE(DATA) TYPE  ZAIF_X102_SAP_FLCUST_CHANGE
*"     REFERENCE(CURR_LINE)
*"     REFERENCE(SUCCESS) TYPE  /AIF/SUCCESSFLAG
*"     REFERENCE(OLD_MESSAGES) TYPE  /AIF/BAL_T_MSG
*"----------------------------------------------------------------------

  CALL FUNCTION 'BAPI_FLCUST_CHANGE'
    EXPORTING
      customernumber  = data-customernumber
      customer_data   = data-customer_data
      customer_data_x = data-customer_data_x
    TABLES
      return          = return_tab.

ENDFUNCTION.
```

Test Interface

Now you can test your interface by executing the report. Select a customer number and enter new data into the other input fields. Execute the report.



Program ZAIF_TEST_X102_CHANGE_FLCUST

Customer Number	00003356
Customer name	Mary Lautenberg
Street	102 Baker Street
Postal Code	W1U 6RN
City	London
Country Indic.	GB
Tel. no.	020 45334 572045
E-Mail Address	mary.laut@home.uk

Figure 1: Report for Changing a Customer

Call transaction /AIF/ERR and select your interface. Select all status. You should see all messages that you triggered with your report. In case errors occurred during processing you can try to edit, restart and cancel the messages. If your message was processed successfully you can check transaction BC_GLOBAL_SCUST_DISP to see the updated customer data.

Processing of Multiple Messages

The enabler for XML message provides you with the possibility to trigger the processing of multiple messages for a specific raw data structure. You have to call static method TRANSFER_TO_AIF_MULT of class /AIF/CL_ENABLER_XML. The method has one mandatory parameter IT_ANY_STRUCTURE. All other parameters are optional. You have to pass a table that has the type of your raw data structure as line type to IT_ANY_STRUCTURE. The method will then trigger processing of a message for each line in IT_ANY_STRUCTURE.

RUNTIME CONFIGURATION GROUP

Messages that are processed with /AIF/CL_ENABLER_XML=>TRANSFER_TO_AIF or /AIF/CL_ENABLER_XML=>TRANSFER_TO_AIF_MULTI are processed with the SAP Application Interface Framework's own runtime. The runtime environment will create persistence runs. Within the runs the messages will be processed by run packs (see Figure 1). The number of runs and the number of run packets that will be created depends on the *Runtime Configuration Group*.

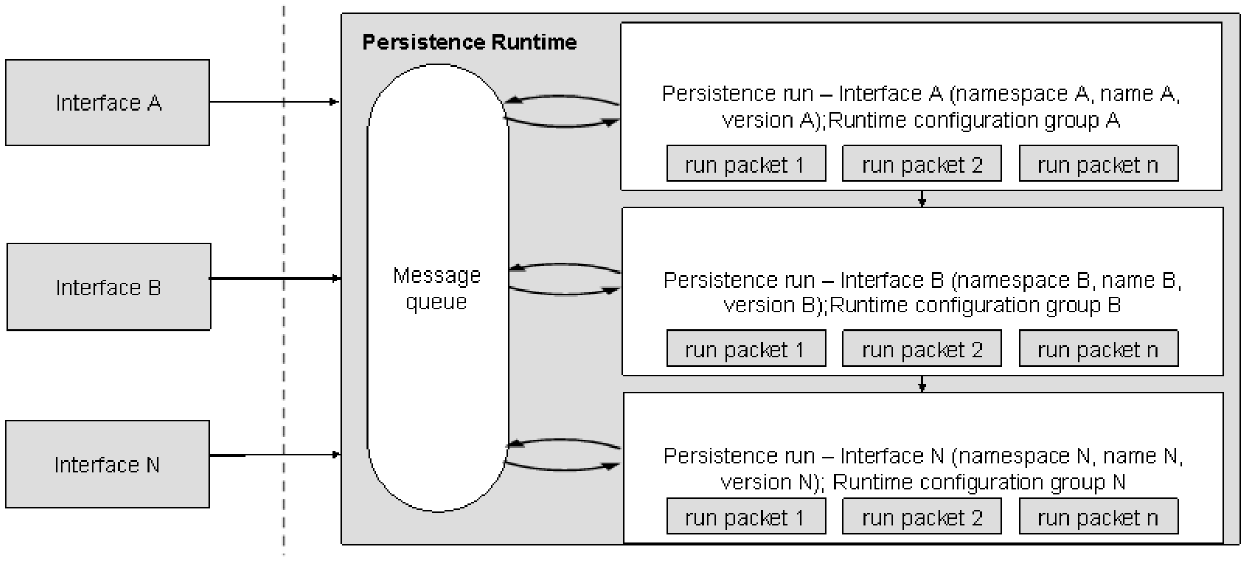

Interface A
Interface B
Interface N
Persistence Runtime
Message queue
Persistence run – Interface A (namespace A, name A, version A);Runtime configuration group A
run packet 1
run packet 2
run packet n
Persistence run – Interface B (namespace B, name B, version B);Runtime configuration group B
run packet 1
run packet 2
run packet n
Persistence run – Interface N (namespace N, name N, version N); Runtime configuration group N
run packet 1
run packet 2
run packet n

Figure 2: Persistence Run

There is a default configuration. However, the runtime of the SAP Application Interface Framework provides you with the possibility to configure how messages should be processed. In order to do so you can define your own *Runtime Configuration Groups* (transaction /AIF/PERS_CGR).

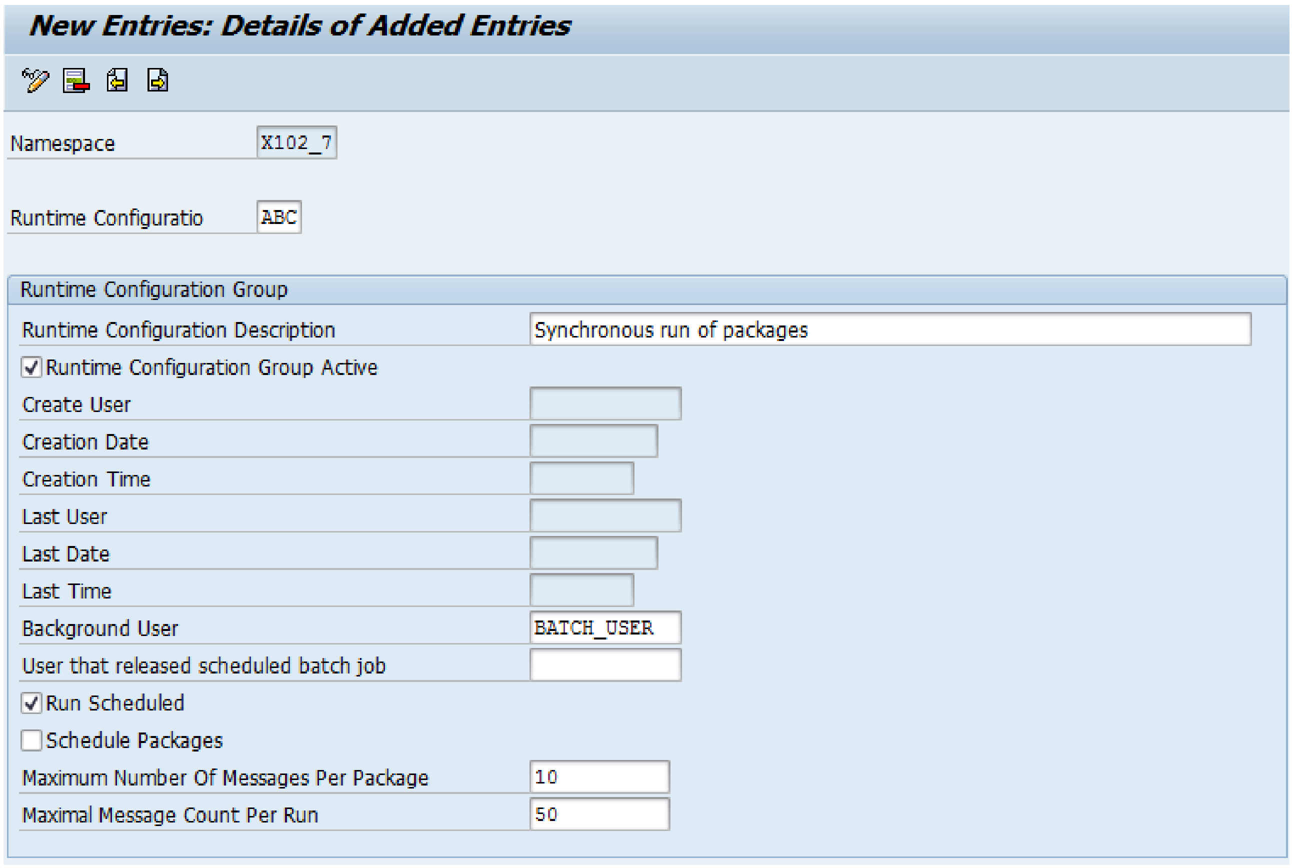
The runtime configuration groups are identified by a namespace and a three character identifier. It is possible to activate or deactivate the runtime configuration group. If the runtime configuration group is deactivated, the messages will be stored in the message queue, but they will not be processed.

Background User field enables you to specify a user that will execute the batch job. This user requires the necessary authorizations to process messages in the SAP Application Interface Framework. If you do not specify a user the current user (SY-UNAME) will be used to release and execute the jobs.

If the *Run Scheduled* indicator is set, the runs are executed asynchronously in a job. Otherwise, the run is executed synchronously.

The *Scheduled Packages* indicator defines if the packets should be executed asynchronously in a job or synchronously. If the indicator is set the packet will be executed asynchronously. If they are executed asynchronously the packets will be scheduled in additional second jobs. Otherwise, the packets will be executed in the same job as the run.

Additionally, you can specify the number of messages that should be executed per packet and the number of messages that should executed per run.



New Entries: Details of Added Entries

Namespace	X102_7
Runtime Configuratio	ABC

Runtime Configuration Group

Runtime Configuration Description	Synchronous run of packages
✓ Runtime Configuration Group Active	
Create User	
Creation Date	
Creation Time	
Last User	
Last Date	
Last Time	
Background User	BATCH_USER
User that released scheduled batch job	
✓ Run Scheduled	
☐ Schedule Packages	
Maximum Number Of Messages Per Package	10
Maximal Message Count Per Run	50

Figure 3: Runtime Configuration Group

Figure 3: Runtime Configuration GroupFigure 3 shows a runtime configuration group ABC in namespace X102_7. Data will be processed with user BATCH_USER. Runs can contain up to 50 messages each. The runs will be created asynchronously. Within a run the packages will be executed synchronously. Within each package up to 10 messages can be processed.

Example 1: When you have 50 messages there will be one run. This run will contain 5 packages with 10 messages each.

Example 2: 51 messages should be processed. Now there will be two runs. The first run will be scheduled and it will contain 5 packages that contain 10 messages each. The second run will be scheduled and will contain 1 package that will contain 1 message.

Methods TRANSFER_TO_AIF and TRANSFER_TO_AIF_MULTI of class /AIF/CL_ENABLER_PROXY have an optional parameter IV_QUEUE_NS and an optional parameter IV_QUEUE_NAME. If you want to use your own runtime configuration group you have to pass the namespace and the three character identifier to those parameters (see code snippet below).

```
/aif/cl_enabler_xml=>transfer_to_aif(
  EXPORTING
    is_any_structure = ls_input
    iv_queue_ns      = 'X102_7'
    iv_queue_name    = 'ABC' ).
```

The default configuration delivered with the SAP Application Interface Framework will process the message in the following way:

- Background job execution user is SY-UNAME
- 20 messages are processed per message packet
- 100 messages are managed per run

www.sap.com