

# SAP HANA SPS 09 - What's New?

## XS Programming Model

(Delta from SPS 08 to SPS 09)

SAP HANA Product Management

November, 2014



# Agenda

---

Miscellaneous Improvements

Security

- User Handling

Outbound HTTP

New Database Interface

Core XSJS APIs

- SMTP
- ZIP

Job Scheduling

OData

Web Dispatcher

XS Admin

- Self Service
- Logon Screen

HANA Test Tools

CDS

- Lifecycle Management

XSDS

- Managed
- Unmanaged
- Procedures

Repository REST APIs



# Miscellaneous Improvements

# HANA XS SPS9: Miscellaneous Improvements

---

**New Mozilla VM (currently 28)**

**Relaxed strict mode settings of the JS VM**

**New Thread Model**

**OData execution tracking (SP9: GET only)**

**Usage Statistics**



# Security Features

# HANA XS SPS9: Misc. Security Features

---

**Full CORS support (Cross-Origin Resource Sharing)**

**Secure http session cookies**

**Support for Client Certificates from F5's BigIP**

**Custom Headers/X-Frame**

**SAML Single Logout (SLO)**

**SAML Authentication in Authorization header**

**Support of Virus Scan Interface (VSI) for applications**

# HANA XS SPS9: Misc. Security Features – CORS Support

## Full CORS support (Cross-Origin Resource Sharing)

- Filtering by origins, headers, and methods

The screenshot shows the SAP HANA XS SPS9 runtime configuration interface. On the left, the Application Objects tree view shows various packages like Wile, playground, and sp9. The main panel displays Runtime Configuration Details for package playground.sp9, delivery unit PLAYGROUND, and source system ORG. The Security & Authentication tab is selected, showing the CORS section. The 'Enable Cross Origin Resource Sharing' checkbox is checked. Under 'Allowed Origins', there is a single entry '\*' with buttons for Add, Edit, and Delete. Under 'Allowed Headers', 'Exposed Headers', and 'Allowed Methods', no entries are present. The 'Max Age' field is set to 3600. At the bottom, there are Save, Cancel, Reset, and Print buttons.

# HANA XS SPS9: Misc. Security Features – Custom Headers/X-Frame

## Custom Headers/X-Frame

- **Controls if the browser should allow a page to be rendered in a frame, iframe, or object.**
- **Avoids *clickjacking* attacks by keeping content from being embedded in a malicious site**

Runtime Configuration Details

Package: playground.sp9      Delivery Unit: PLAYGROUND  
Source System: ORG

Security & Authentication    CORS    **Custom Headers**

Enable Custom Headers

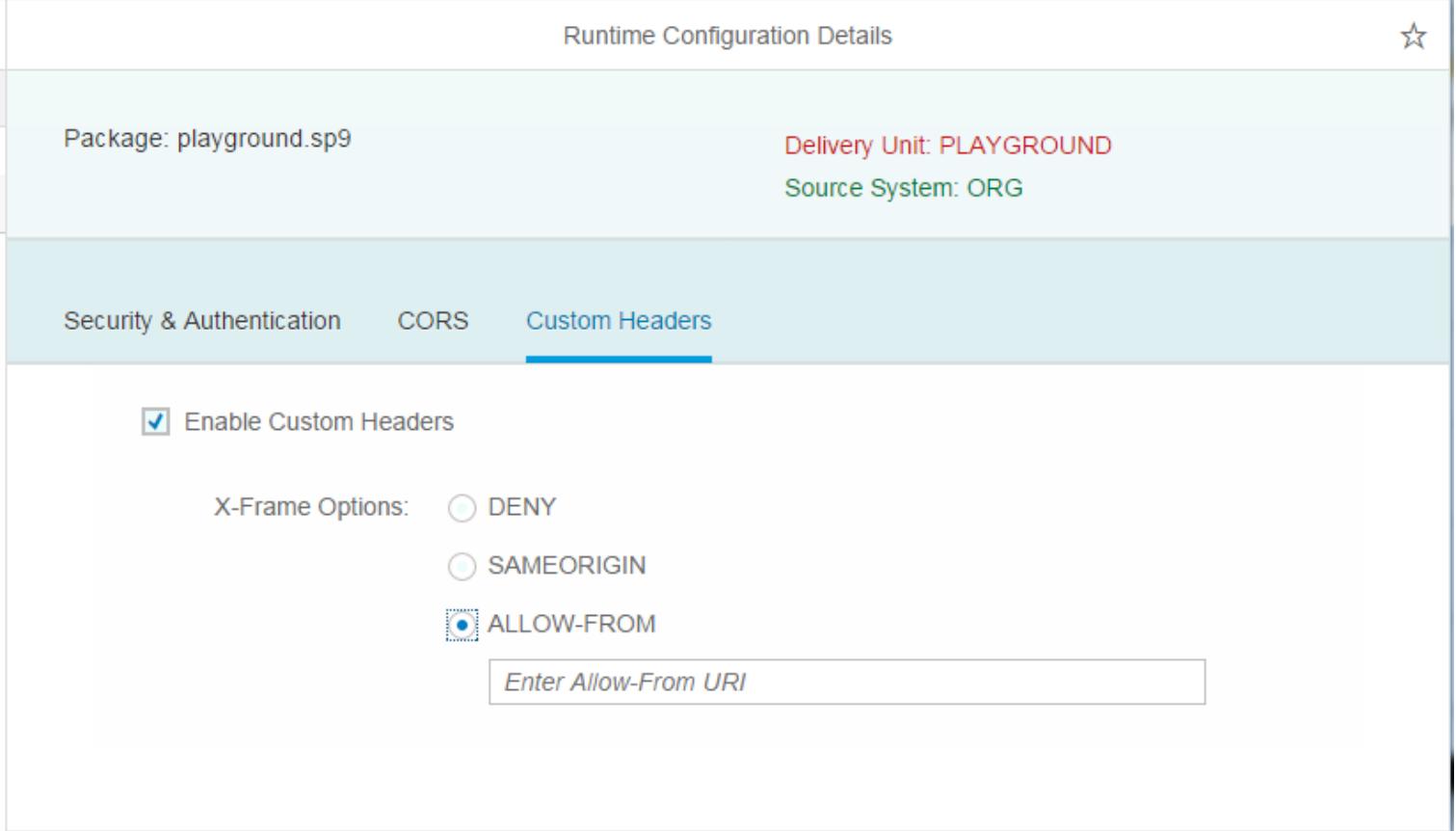
X-Frame Options:

DENY

SAMEORIGIN

ALLOW-FROM

*Enter Allow-From URI*





# User Handling

# HANA XS SPS9: Recommended User Handling

---

## **Recommended default pattern: no SQL privileges for end-users!**

Named DB users are used for logon and assignment of XS application privileges

Default connection is used to switch to technical user for all DB access

- Access to name/ID of logged-on user still possible
- Instance filtering is still done for the logged-on user

## **Improved Support for Close Cooperation with ABAP Applications**

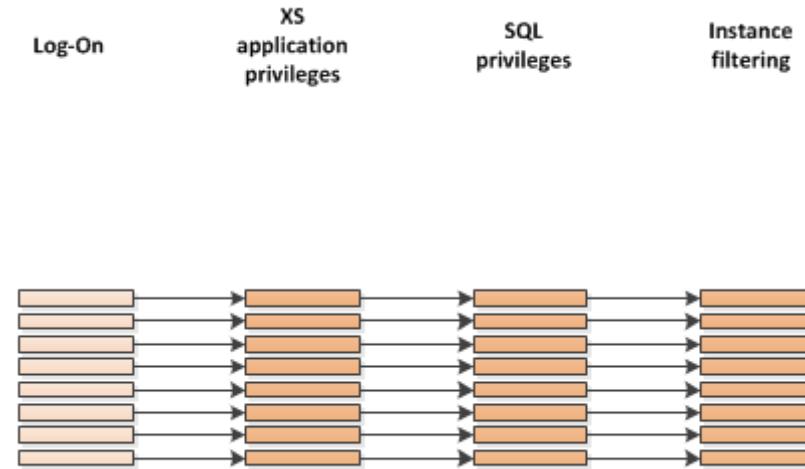
Logon via SAML or SAP Logon Tickets is possible even without named DB User!

Programmatic alternative (SqlScript) to XS application privileges

# User handling in XS

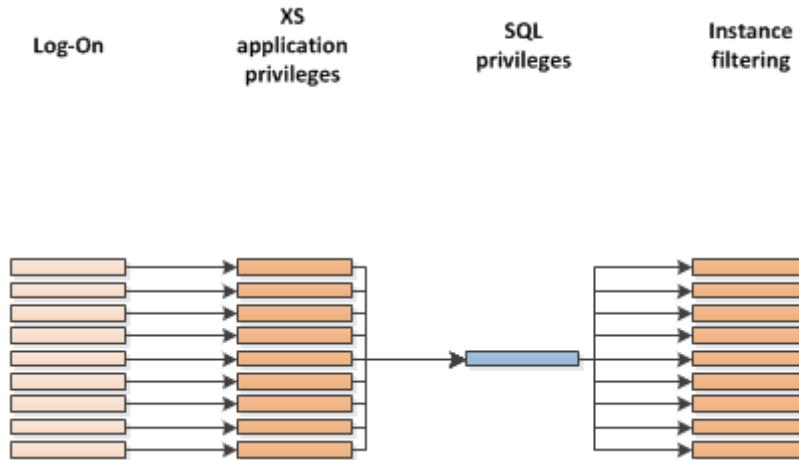
## Plain DB user

- The logon maps to the personal DB user, who is used for checking XS application privileges
- SQL connections are opened for the personal DB user, who is thus also used for checking SQL privileges
- Instance-filtering is also done with personal DB user
- Personal DB user needs SQL privileges which he can abuse via SQL Console/Studio



# User handling in XS SQLCC scenario (best practice for stand-alone XS Apps)

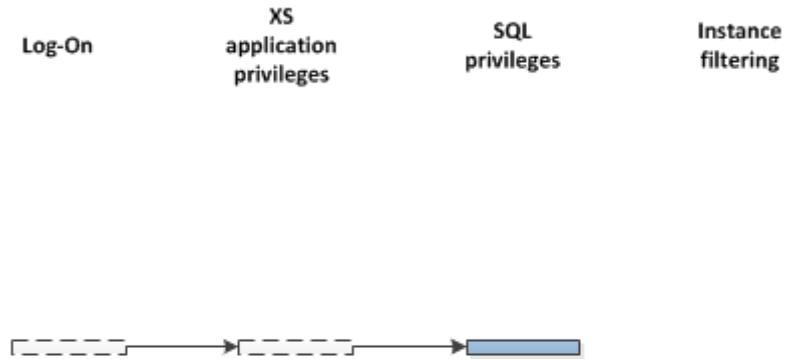
- **The logon maps to the personal DB user**
- **XS privilege checks are done with this user**
- **Package without configured default connection**
  - OData services and plain DB access from XSJS are not recommended, to avoid granting SQL privileges to personal DB users
  - Controlled navigation to non-exposed packages which have a configured default connection
- **Packages with configured default connection**
  - All SQL connections (xsjs and OData) are opened for the configured default connection, which is thus used for checking all SQL privileges
- **Instance-filtering is not possible (planned for SPS09)**  
**Personal DB user needs only XS Privileges**



# User handling in XS

## Anonymous section scenario

- **No logon is enforced**
- **XS privilege checks will fail**
- **Package without configured default connection**
  - OData services and plain DB access from XSJS are not possible
  - navigation to packages with XS privilege check is not possible
- **Packages with configured default connection**
  - All SQL connections (xsjs and OData) are opened for the configured default connection, which is thus used for checking all SQL privileges
- **Instance-filtering is not possible**

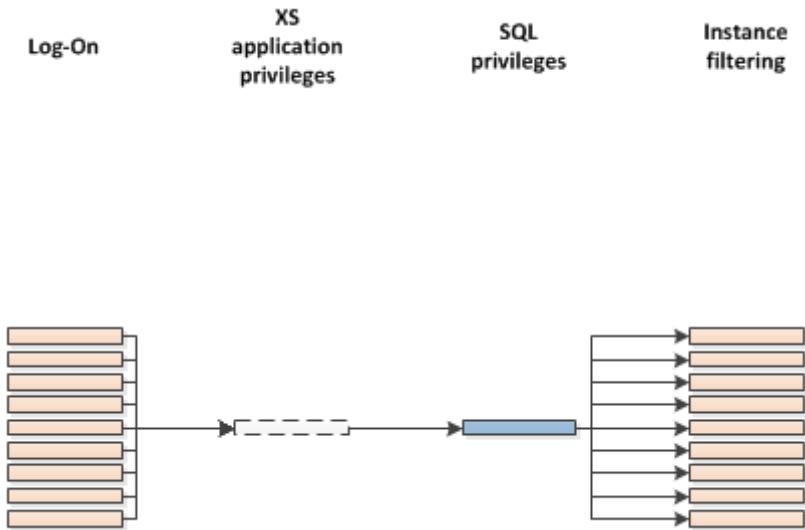


# User handling in XS

## Technical user scenario

New scenario in  
SPS09

- **The logon is successful without mapping to a personal DB user**
- **XS privilege checks will fail**
- **Package without configured default connection**
  - DB access not possible
- **Packages with configured default connection**
  - All SQL connections (xsjs and OData) are opened for the configured sqlcc user, which is thus used for checking all SQL privileges
- **Instance-filtering is done with logon user**





# Outbound Connectivity

# HANA XS SPS9: Additional features for Outbound Connectivity

---

**Anonymous Outbound SSL supported**

**Act as SAML 2.0 ID Provider**

**OAuth 2.0 client**

**Option to Enforce Proxy for all outgoing communication**



# New Database Interface

# New database interface

---

## Performance gain

- Achieves higher throughput
- Better scale out support
- Minimal remote traffic

## Usability improvements

- Simple and easy to use JavaScript interface
- Code reduction
- No boilerplate code

## Light-weight architecture

- Based upon thin C++ client library
- No SQL Processing in XS layer
- Uses internal HANA communication protocol

# New database interface

## Examples – old \$.db interface

---

```
var conn = $.db.getConnection();

var query = 'SELECT * FROM "sap.hana.democontent.epmNext.data::EPM.Purchase.Item" ' +
    ' WHERE "PRODUCT.PRODUCTID" = ? ';

var pstmt = conn.prepareStatement(query);
pstmt.setString(1, productId);

var rs = pstmt.executeQuery();

var body = '';
while (rs.next()) {
    var gross = rs.getDecimal(6);
    if(gross >= 500){
        body += rs.getString(1) + "\t" + rs.getString(2) + "\t" +
            rs.getString(3) + "\t" + rs.getDecimal(6) + "\n"; }
}
rs.close();
pstmt.close();
```

# New database interface

## Examples – old \$.db interface

```
var conn = $.db.getConnection();

var query = 'SELECT * FROM "sap.hana.democontent.epmNextGen"
             ' WHERE "PRODUCT.PRODUCTID" = ? ';

var pstmt = conn.prepareStatement(query);
pstmt.setString(1, productId);

var rs = pstmt.executeQuery();

var body = '';
while (rs.next()) {
    var gross = rs.getDecimal(6);
    if(gross >= 500){
        body += rs.getString(1) + "\t" + rs.getString(2) + "\t" +
               rs.getString(3) + "\t" + rs.getDecimal(6) + "\n"; }
}
rs.close();
pstmt.close();
```

Parameters set via  
separate command –  
type specific

# New database interface

## Examples – old \$.db interface

```
var conn = $.db.getConnection();

var query = 'SELECT * FROM "sap.hana.democontent.epmNext.data::EPM.Purchase.Item" ' +
            ' WHERE "PRODUCT.PRODUCTID" = ? ';

var pstmt = conn.prepareStatement(query);
pstmt.setString(1, productId);

var rs = pstmt.executeQuery();

var body = '';
while (rs.next()) {
    var gross = rs.getDecimal(6);
    if(gross >= 500){
        body += rs.getString(1) + "\t" + rs.getString(2) + "\t" +
               rs.getString(3) + "\t" + rs.getDecimal(6) + "\n";
    }
}
rs.close();
pstmt.close();
```



Returned Record Set –  
no random access,  
one loop sequential

# New database interface

## Examples – old \$.db interface

```
var conn = $.db.getConnection();

var query = 'SELECT * FROM "sap.hana.democontent.epmNext.data::EPM.Purchase.Item" ' +
            ' WHERE "PRODUCT.PRODUCTID" = ? ';

var pstmt = conn.prepareStatement(query);
pstmt.setString(1, productId);

var rs = pstmt.executeQuery();

var body = '';
while (rs.next()) {
    var gross = rs.getDecimal(6);
    if(gross >= 500){
        body += rs.getString(1) + "\t" + rs.getString(2) + "\t" +
               rs.getString(3) + "\t" + rs.getDecimal(6) + "\n";
    }
}
rs.close();
pstmt.close();
```

Each cell must be read  
with another type  
specific function call

# New database interface

## Examples – new \$.hdb interface

```
var conn = $.hdb.getConnection();

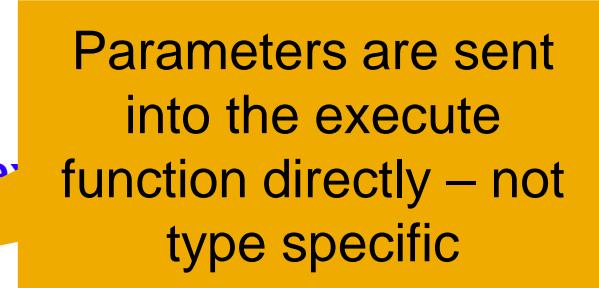
var query = 'SELECT * FROM "sap.hana.democontent.epmNext.data::EPM.Purchase.Item"' +
            ' WHERE "PRODUCT.PRODUCTID" = ?';
var rs = conn.executeQuery(query,productId);

var body = '';
for(var i = 0; i < rs.length; i++){
    if(rs[i]["GROSSAMOUNT"] >= 500){
        body += rs[i]["HEADER.PURCHASEORDERID"] + "\t" + rs[i]["PURCHASEORDERITEM"] +
                "\t" + rs[i]["PRODUCT.PRODUCTID"] + "\t" + rs[i]["GROSSAMOUNT"] + "\n";
    }
}
```

# New database interface

## Examples – new \$.hdb interface

```
var conn = $.hdb.getConnection();  
  
var query = 'SELECT * FROM "sap.hana.democontent.epmNe...  
    ' WHERE "PRODUCT.PRODUCTID" = ?';  
var rs = conn.executeQuery(query,productId);  
  
var body = '';  
for(var i = 0; i < rs.length; i++){  
    if(rs[i]["GROSSAMOUNT"] >= 500){  
        body += rs[i]["HEADER.PURCHASEORDERID"] + "\t" + rs[i]["PURCHASEORDERITEM"] +  
        "\t" + rs[i]["PRODUCT.PRODUCTID"] + "\t" + rs[i]["GROSSAMOUNT"] + "\n";  
    }  
}
```



Parameters are sent into the execute function directly – not `item` + type specific

# New database interface

## Examples – new \$.hdb interface

```
var conn = $.hdb.getConnection();

var query = 'SELECT * FROM "sap.hana.democonsignment"
            ' WHERE "PRODUCT.PRODUCTID" = ?';
var rs = conn.executeQuery(query,productId);

var body = '';
for(var i = 0; i < rs.length; i++){
    if(rs[i]["GROSSAMOUNT"] >= 500){
        body += rs[i]["HEADER.PURCHASEORDERID"] + "\t" + rs[i]["PURCHASEORDERITEM"] +
        "\t" + rs[i]["PRODUCT.PRODUCTID"] + "\t" + rs[i]["GROSSAMOUNT"] + "\n";
    }
}
```

Returned Record Set is a JSON object – direct index access or easy looping

1. Purchase.Item" +

# New database interface

## Examples – new \$.hdb interface

```
var conn = $.hdb.getConnection();

var query = 'SELECT * FROM "sap.hana.democontent.epi"
            ' WHERE "PRODUCT.PRODUCTID" = ?';
var rs = conn.executeQuery(query,productId);

var body = '';
for(var i = 0; i < rs.length; i++){
    if(rs[i]["GROSSAMOUNT"] >= 500){
        body += rs[i]["HEADER.PURCHASEORDERID"] + "\t" + rs[i]["PURCHASEORDERITEM"] +
        "\t" + rs[i]["PRODUCT.PRODUCTID"] + "\t" + rs[i]["GROSSAMOUNT"] + "\n";
    }
}
```

Record Set object doesn't  
materialize the data into  
JavaScript VM

+

# New database interface

## Examples – new \$.hdb interface

```
var conn = $.hdb.getConnection();

var query = 'SELECT * FROM "sap.hana.democontent.epmNext_data::FDM_Purchase.Item"' +
            ' WHERE "PRODUCT.PRODUCTID" = ?';
var rs = conn.executeQuery(query,productId);

var body = '';
for(var i = 0; i < rs.length; i++){
    if(rs[i]["GROSSAMOUNT"] >= 500){
        body += rs[i]["HEADER.PURCHASEORDERID"] + "\t" + rs[i]["PURCHASEORDERITEM"] +
                "\t" + rs[i]["PRODUCT.PRODUCTID"] + "\t" + rs[i]["GROSSAMOUNT"] + "\n";
    }
}
```

Individual Cells  
accessible by name /  
not type specific

# New database interface

## Debugging

New interface makes debugging easier as the result set is JSON and fully accessible in the debugger

The screenshot shows the SAP HANA Studio Debugger interface. On the left is a code editor window titled "largeAfter.xsjs" containing the following JavaScript code:

```
1 var conn = $.hdb.getConnection();
2
3 var query = 'SELECT * FROM "SAP_HANA_EPM_NEXT"."sap.hana.democontent.epmNext"';
4 var rs = conn.executeQuery(query);
5
6 for(var i = 0; i < rs.length; i++) {
7
8     var poId = rs[i]["HEADER.PURCHASEORDERID"];
9     var product = rs[i]["PRODUCT.PRODUCTID"];
10    //Some processing
11    if(product==='TEST') {}
12 }
13
```

The line number 11 is highlighted with a yellow background. The "Watch Area" panel on the right displays the result set of the query, which is a JSON object representing a row from the database:

- query: "SELECT \* FROM \"SAP\_HANA\_EPM\_NEXT\".\"sap.hana.democontent.epmNext"
- rs:[object ResultSet]
  - 0:[object Row]
    - CURRENCY:'EUR'
    - DELIVERYDATE:[object Date]
    - GROSSAMOUNT:'1137.64'
    - HEADER.PURCHASEORDERID:'0300000000'
    - NETAMOUNT:'956.00'
    - NOTEID:null
    - PRODUCT.PRODUCTID:'HT-1000'
    - proto:[object Row]
    - PURCHASEORDERITEM:'0000000010'
    - QUANTITY:'1.000'
    - QUANTITYUNIT:'EA'
    - TAXAMOUNT:'181.64'
    - Functions
  - 1000:[object Row]
    - CURRENCY:'ARS'
    - DELIVERYDATE:[object Date]

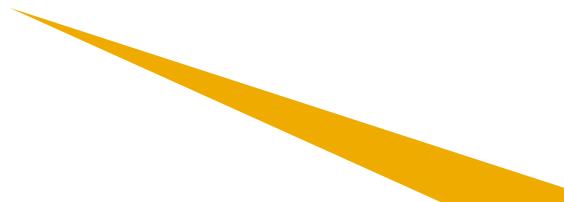
Below the Watch Area are the Call Stack and Breakpoints panels.

# New database interface

## Examples – new \$.hdb interface, JSON RecordSet

```
var connection = $.hdb.getConnection();
var results = connection.executeQuery(
'SELECT * FROM "sap.hana.democontent.epmNext.data::EPM.MasterData.Employees" ' +
'WHERE LOGINNAME <> ?', 'EPM_USER');

$.response.setBody(JSON.stringify(results));
```



Result Record Set is  
JSON; ready for output

# New database interface

## Examples – new \$.hdb interface, Stored Procedures

```
var connection = $.hdb.getConnection();

var partnerRole = $.request.parameters.get("PartnerRole");
partnerRole = typeof partnerRole !== 'undefined' ? partnerRole : '01';

var getBpAddressesByRole = connection.loadProcedure("SAP HANA EPM NEXT",
"sap.hana.democontent.epmNext.procedures::get_bp_addresses_by_role");

var results = getBpAddressesByRole(partnerRole);

//Pass output to response
$.response.status = $.net.http.OK;
$.response.contentType = "application/json";
$.response.setBody(JSON.stringify(results));
```

Procedure has a  
JavaScript function  
“proxy”

# New database interface

## Examples – new \$.hdb interface, Stored Procedures

```
var connection = $.hdb.getConnection();

var partnerRole = $.request.parameters.get("PartnerRole");
partnerRole = typeof partnerRole !== 'undefined' ? partnerRole : '01';

var getBpAddressesByRole = connection.loadProcedure("SAP_HANA_EPM_NEXT",
"sap.hana.democontent.epmNext.procedures::get_bp_addresses_by_role");

var results = getBpAddressesByRole(partnerRole);

//Pass output to response
$.response.status = $.net.http.OK;
$.response.contentType = "application/json";
$.response.setBody(JSON.stringify(results));
```

JavaScript input  
parameters / Table  
parameters as JSON

# New database interface

## Examples – new \$.hdb interface, Stored Procedures

```
var connection = $.hdb.getConnection();

var partnerRole = $.request.parameters.get("PartnerRole");
partnerRole = typeof partnerRole !== 'undefined' ? partnerRole : '01';

var getBpAddressesByRole = connection.loadProcedure("SAP_HANA_EPM_NEXT",
"sap.hana.democontent.epmNext.procedures::get_bp_addresses_by_role");

var results = getBpAddressesByRole(partnerRole);

//Pass output to response
$.response.status = $.net.http.OK;
$.response.contentType = "application/json";
$.response.setBody(JSON.stringify(results));
```

Result Record Set is  
also JSON



# New Core XSJS APIs

# HANA XS SPS9: New XSJS-APIs

---

<b>SMTP (sending mails from xsjs application code)</b>	<b>Text Access</b>
<b>ZIP and GZIP support (read and write)</b>	<b>Text Mining</b>
<b>XML parsing support</b>	<b>Management of Scheduled Jobs</b>
<b>Secure Store for applications</b>	<b>Various utility functions (hash, encrypt)</b>
<b>Better Multipart Support</b>	<b>AntiVirus for applications</b>
<b>Asynchronous request completion</b>	

# New XSJS-APIs

## Examples – SMTP

---

```
var mail = new $.net.Mail({
    sender: {address: "demo@sap.com"},
    to: [{ address: "user@sap.com"}],
    subject: "XSJS Email Test",
    parts: [ new $.net.Mail.Part({
        type: $.net.Mail.Part.TYPE_TEXT,
        text: "The body of the mail.",
        contentType: "text/plain"
    })]
});
var returnValue = mail.send();
var response = "MessageId = " + returnValue.messageId +
    ", final reply = " + returnValue.finalReply;
```

# New XSJS-APIs

## Examples – SMTP with attachment

---

```
var mail = new $.net.Mail({
    sender: {address: "demo@sap.com"},
    to: [{ address: "user@sap.com"}],
    subject: "XSJS Email Test",
    parts: [ new $.net.Mail.Part({
        type: $.net.Mail.Part.TYPE_TEXT,
        text: "Atachement Test",
        contentType: "text/plain"
    })]
});
mail.parts.push(new $.net.Mail.Part({
    type: $.net.Mail.Part.TYPE_ATTACHMENT,
    data: getImage(),
    contentType: "image/jpg",
    fileName: "myPicture.jpg"}));
var returnValue = mail.send();
```

# New XSJS-APIs

## Examples – SMTP Configuration

The screenshot shows the SAP XS Artifitct Administration interface. The left sidebar has a navigation menu with items: XS Artifact Administration, SAML Service Provider, SAML Identity Provider, Trust Manager, **SMTP Configurations**, and XS Job Dashboard. The main area is titled "SMTP Configurations". It contains three sections: "General SMTP Settings", "Transport Security Settings", and "Socket Proxy Settings". In the "General SMTP Settings" section, "Mail Server Host" is set to "mail.corp" and "Mail Server Port" is set to "25". In the "Transport Security Settings" section, "Transport Secur..." is set to "None" and "Trust Store" is set to "None". In the "Socket Proxy Settings" section, "SOCKS Proxy" is set to "OFF", and "Proxy Host", "Proxy Port", "Proxy Username", and "Proxy Password" fields are all empty. On the right side, there is an "Authentication" section with a dropdown menu. The dropdown menu is open, showing options: None (selected), None, Auto, Plain, Login, CRAM-MD5, and Digest-MD5. A cursor arrow points to the "None" option in the dropdown.

XS Artifact Administration

SAML Service Provider

SAML Identity Provider

Trust Manager

**SMTP Configurations**

XS Job Dashboard

SMTP Configurations

General SMTP Settings

Mail Server Host: mail.corp

Mail Server Port: 25

Transport Security Settings

Transport Secur...: None

Trust Store: None

Socket Proxy Settings

SOCKS Proxy: OFF

Proxy Host: Enter Socket Proxy Host

Proxy Port: 1080

Proxy Username: Enter Socket Proxy Username

Proxy Password: Enter Socket Proxy Password

Authentication

Authentication T...:

- None
- None
- Auto
- Plain
- Login
- CRAM-MD5
- Digest-MD5

# New XSJS-APIs

## Examples – ZIP

---

```
var zip = new $.util.Zip();
zip["folder1/demo1.txt"] = "This is the new ZIP Processing in XSJS";
zip["demo2.txt"] = "This is also the new ZIP Processing in XSJS";

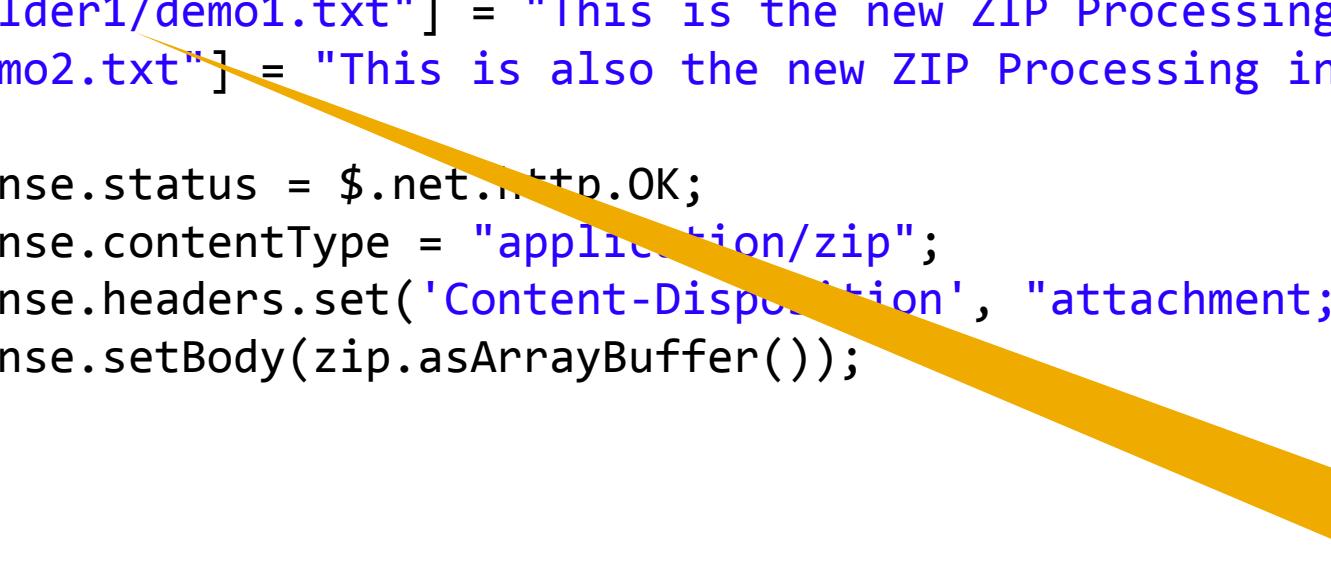
$.response.status = $.net.http.OK;
$.response.contentType = "application/zip";
$.response.headers.set('Content-Disposition', "attachment; filename = 'ZipExample.zip'");
$.response.setBody(zip.asArrayBuffer());
```

# New XSJS-APIs

## Examples – ZIP

```
var zip = new $.util.Zip();
zip["folder1/demo1.txt"] = "This is the new ZIP Processing in XSJS";
zip["demo2.txt"] = "This is also the new ZIP Processing in XSJS";

$.response.status = $.net.http.OK;
$.response.contentType = "application/zip";
$.response.headers.set('Content-Disposition', "attachment; filename = 'ZipExample.zip'");
$.response.setBody(zip.asArrayBuffer());
```



Folder structure in ZIP  
simply by specify the  
full path as you create  
the content

# New XSJS-APIs

## Examples – ZIP Continued

---

**Load Zip directly from web request object – avoid moving content into JavaScript variable**

```
var zip = new $.util.Zip($.request.body);
```

**Similar direct access for Record Set result object of a database query**

```
statement = conn.prepareStatement("select data from EXAMPLETABLE where id=1");
var rs = statement.executeQuery();
if (rs) {
    while (rs.next()) {
        //Load Zip From ResultSet
        var loadedZip = new $.util.Zip(rs, 1);
    }
}
```

# New XSJS-APIs

## Examples – XML

```
var parser = new $.util.SAXParser();

//parse XML from String
var parser = new $.util.SAXParser();
var xml = '<?xml version="1.0" encoding="UTF-8" standalone="yes"?>\n' +
    '<!-- this is a note -->\n'+<note noteName="NoteName">' +
    '<to>To</to>'+<from>From</from>'+<heading>Note heading</heading>' +
    '<body>Note body</body>'+</note>';
var startElementHandlerConcat = "";

parser.startElementHandler = function(name, attrs) {
    startElementHandlerConcat += name;
    if (name === "note") {
        startElementHandlerConcat += " noteName = '" + attrs.noteName + "'";
    }
    startElementHandlerConcat += "\n";
};

parser.parse(xml);
```

# New XSJS-APIs

## Examples – Improved Multi-Part

---

```
var i;
var n = $.request.entities.length;
var client = new $.net.http.Client();
for (i = 0; i < n; ++i) {
    var childRequest = $.request.entities[i].body.asWebRequest();
    client.request(childRequest, childRequest.headers.get("Host") + childRequest.path);
    var childResponse = client.getResponse();
    var responseEntity = $.response.entities.create();
    responseEntity.setBody(childResponse);
}
```

# New XSJS-APIs

## Examples – Response Callback

---

```
$.response.followUp({  
    uri : "playground.sp9.followUp:other.xsjs",  
    functionName : "doSomething",  
    parameter : {  
        a : "b"  
    }  
});
```

# New XSJS-APIs

## Examples – Text Access

---

```
var textAccess = $.import("sap.hana.xs.i18n","text");
var bundle = textAccess.loadBundle("playground.sp9.textAccess","demo1");
var singleText = bundle.getText("demo");
var replaceText = bundle.getText("demo2",['1001']);
var oAllTexts = bundle.getTexts();
```

# New XSJS-APIs

## Examples – AntiVirus

```
try {
    //create a new $.security.AntiVirus object using the default profile
    var av = new $.security.AntiVirus();
    av.scan($.request.body);
} catch (e) {
    $.response.setBody(e.toString());
}
```

- The scan needs a Virus Scan Adapter (VSA) to be installed on the host
- The setup and configuration is available with SAP note 2081108
- This class uses the SAP certified interface NW-VSI 2.00 (see SAP note 1883424)
- For a list of the AV products supported, see SAP note 1494278

# New XSJS-APIs

## Examples – Secure Store

```
function store() {  
    var config = {  
        name: "foo",  
        value: "bar"  
    };  
    var aStore = new $.security.Store("localStore.xssecurestore");  
    aStore.store(config);  
}  
  
function read() {  
    var config = {  
        name: "foo"  
    };  
    try {  
        var store = new $.security.Store("localStore.xssecurestore");  
        var value = store.read(config);  
    }  
    catch(ex) {  
        //do some error handling  
    }  
}
```



# Job Scheduling

# HANA XS SPS9: Scheduled Jobs

---

## Robustness

- Active jobs will not block DU import anymore
- No hick-up of scheduling, even in case of process crashes

## Enhanced xsjs API for Job Management

- Enumerate schedules of a job object (access as e.g. myjob.schedules[19])
- Modify schedules via properties (xscron, description, parameter like myjob.schedules[19].xscron = “\* \* \* \* \*”;)
- Expose job\_log entries of a schedule as a property (like myjob.schedules[19].logs.jobLog)

## Enhanced support for SQLScript parameters

- Scalar parameters
- References to tables
- Creating local tables by type

## New Management UI (XS Job Dashboard)

# HANA XS SPS9: Scheduled Jobs – New Management UI

## New Management UI (XS Job Dashboard)

URL: /sap/hana/xs/admin/jobs/

Name	Package	User	Status	Start Time	End Time	Session Timeout	Last Run Status
VDSchedulerJob_1	sap.bi.discover.lo gic.vizd. backgroundjobs		INACTIVE				
VDSchedulerJob_2	sap.bi.discover.lo gic.vizd. backgroundjobs		INACTIVE				
VDSchedulerJob_3	sap.bi.discover.lo gic.vizd. backgroundjobs		INACTIVE				
cleanUp	sap.hana.democontent.epmNextJobs	SYSTEM	ACTIVE			SUCCESS	
cleanUp	sap.hana.xsopen.session.services	SYSTEM	ACTIVE			SUCCESS	
demoSQLScript	playground.sp7.jobs	SYSTEM	INACTIVE			SUCCESS	
demoSQLScript	sap.hana.democontent.epm.jobs		INACTIVE				
demoSQLScript	sap.hana.democontent.epmNextJobs		INACTIVE				
demoXSJS	playground.sp7.jobs	SYSTEM	INACTIVE			SUCCESS	
demoXSJS	sap.hana.democontent.epm.jobs		INACTIVE				
demoXSJS	sap.hana.democontent.epmNextJobs	SYSTEM	INACTIVE			SUCCESS	
schedule	sap.hana.testtools.unit.jasminexs		INACTIVE				

# HANA XS SPS9: Scheduled Jobs – New Management UI Continued

## New Management UI (XS Job Dashboard)

Drill into details of a single job definition from Job Dashboard or from the XS Admin tool

The screenshot shows the SAP HANA XS SPS9 New Management UI. At the top, there's a header with the SAP logo and a user ID (I809764). Below the header, the page title is "XS Job Details". There are two tabs: "Job Details" (which is selected) and "Configuration". Under "Job Details", the job information is displayed:

Name:	cleanUp
Description:	Clean Up Old Cookies
Action:	sap.hana.democontent.epmNext.procedures::clean_up_old_cookies

Below this, the "Schedules" section is shown, listing ten scheduled runs for job ID 56. The columns are: ID, XSCron, Description, Parameter, Planned..., Status, Start..., Finis..., and Time Taken (ms). The status for all entries is "SCHEDULED". The last entry has a cursor icon over it.

ID	XSCron	Description	Parameter	Planned...	Status	Start...	Finis...	Time Taken (ms)
56	*****2 0 0	Will run once a day at 2:00am		2014-10-15...	SCHEDULED			
56	*****2 0 0	Will run once a day at 2:00am		2014-10-14...	SUCCESS	2014-10-14...	2014-10-14...	0
56	*****2 0 0	Will run once a day at 2:00am		2014-10-13...	SUCCESS	2014-10-13...	2014-10-13...	0
56	*****2 0 0	Will run once a day at 2:00am		2014-10-12...	SUCCESS	2014-10-12...	2014-10-12...	0
56	*****2 0 0	Will run once a day at 2:00am		2014-10-11...	SUCCESS	2014-10-11...	2014-10-11...	0
56	*****2 0 0	Will run once a day at 2:00am		2014-10-10...	SUCCESS	2014-10-10...	2014-10-10...	0
56	*****2 0 0	Will run once a day at 2:00am		2014-10-09...	SUCCESS	2014-10-09...	2014-10-09...	0
56	*****2 0 0	Will run once a day at 2:00am		2014-10-08...	SUCCESS	2014-10-08...	2014-10-08...	0
56	*****2 0 0	Will run once a day at 2:00am		2014-10-07...	SUCCESS	2014-10-07...	2014-10-07...	0

At the bottom right of the schedule table, there are buttons for "Add Schedule" and "Delete Schedule".



# OData

# HANA XS SPS9: OData

---

**Configurable Cache-settings for Metadata-Document**

**E-Tag support**

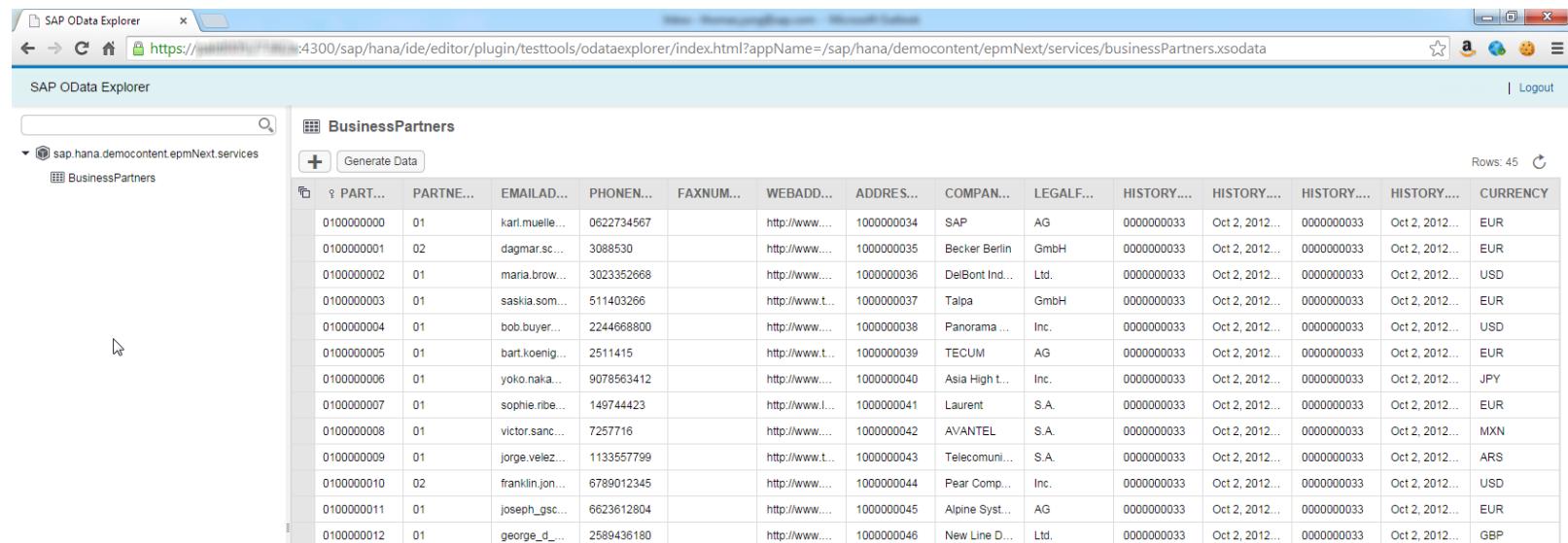
**Automatic parallelization for GET batch requests**

**INA Search capabilities integrated into OData**

# OData Explorer

## SAP River application explorer rebuilt as SAP OData Explorer in SPS09

- General OData test and data generation tool which supports XSODATA services
- Launches from the SAP Web-based Development Workbench or via URL
- `/sap/hana/ide/editor/plugin/testtools/odataexplorer/index.html?appName=<xsodata service path>`



The screenshot shows the SAP OData Explorer interface. The browser address bar displays the URL: `https://<host>:4300/sap/hana/ide/editor/plugin/testtools/odataexplorer/index.html?appName=/sap/hana/democontent/epmNext/services/businessPartners.xsodata`. The main content area is titled "BusinessPartners". It features a search bar and a "Generate Data" button. Below the title, there is a table with 15 rows of data. The columns are labeled: PART..., PARTNE..., EMAILAD..., PHONEN..., FAXNUM..., WEBADD..., ADDRES..., COMPAN..., LEGALF..., HISTORY..., HISTORY..., HISTORY..., HISTORY..., CURRENCY. The data represents various business partners with their respective details like name, address, and contact information.

PART...	PARTNE...	EMAILAD...	PHONEN...	FAXNUM...	WEBADD...	ADDRES...	COMPAN...	LEGALF...	HISTORY...	HISTORY...	HISTORY...	HISTORY...	CURRENCY
0100000000	01	karl.muelle...	0622734567		http://www....	1000000034	SAP	AG	0000000033	Oct 2, 2012...	0000000033	Oct 2, 2012...	EUR
0100000001	02	dagmar.sc...	3088530		http://www....	1000000035	Becker Berlin	GmbH	0000000033	Oct 2, 2012...	0000000033	Oct 2, 2012...	EUR
0100000002	01	maria.brow...	3023352668		http://www....	1000000036	DeBont Ind...	Ltd.	0000000033	Oct 2, 2012...	0000000033	Oct 2, 2012...	USD
0100000003	01	saskia.som...	511403266		http://www.t...	1000000037	Talpa	GmbH	0000000033	Oct 2, 2012...	0000000033	Oct 2, 2012...	EUR
0100000004	01	bob.buyer...	2244668800		http://www....	1000000038	Panorama ...	Inc.	0000000033	Oct 2, 2012...	0000000033	Oct 2, 2012...	USD
0100000005	01	bart.koenig...	2511415		http://www.t...	1000000039	TECUM	AG	0000000033	Oct 2, 2012...	0000000033	Oct 2, 2012...	EUR
0100000006	01	yoko.naka...	9078563412		http://www....	1000000040	Asia High ...	Inc.	0000000033	Oct 2, 2012...	0000000033	Oct 2, 2012...	JPY
0100000007	01	sophie.ribe...	149744423		http://www.l...	1000000041	Laurent	S.A.	0000000033	Oct 2, 2012...	0000000033	Oct 2, 2012...	EUR
0100000008	01	victor.sanc...	7257716		http://www....	1000000042	AVANTEL	S.A.	0000000033	Oct 2, 2012...	0000000033	Oct 2, 2012...	MXN
0100000009	01	jorge.velez...	1133557799		http://www.t...	1000000043	Telecomuni...	S.A.	0000000033	Oct 2, 2012...	0000000033	Oct 2, 2012...	ARS
0100000010	02	franklin.jon...	6789012345		http://www....	1000000044	Pear Comp...	Inc.	0000000033	Oct 2, 2012...	0000000033	Oct 2, 2012...	USD
0100000011	01	joseph_gsc...	6623612804		http://www....	1000000045	Alpine Syst...	AG	0000000033	Oct 2, 2012...	0000000033	Oct 2, 2012...	EUR
0100000012	01	george_d...	2589436180		http://www....	1000000046	New Line D...	Ltd.	0000000033	Oct 2, 2012...	0000000033	Oct 2, 2012...	GBP



# Web Dispatcher

# HANA XS SPS9: Local Web Dispatcher

---

## Full Integration into HANA Process Management

- Monitoring
- Configuration
- Administration

# HANA XS SPS9: Local Web Dispatcher

## Full Integration into HANA Process Management

- Monitoring

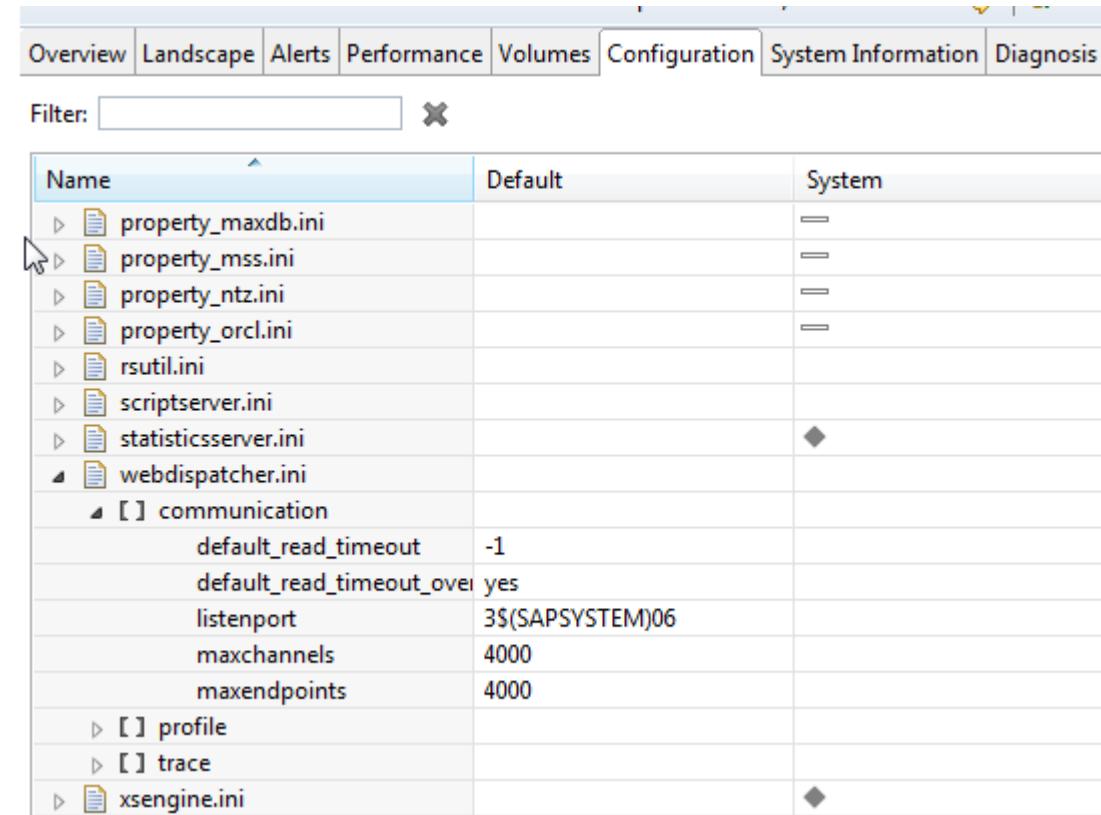
The screenshot shows a monitoring interface for SAP HANA Process Management. The top navigation bar includes tabs for Overview, Landscape, Alerts, Performance, Volumes, Configuration, System Information, Diagnosis Files, Trace Configuration, and Console. Below the navigation is a search bar with fields for Host and Service, and a clear button.

Services	Hosts	Redistribution	System Replication	Host:	<All>	Service:	<All>	X		
Active	Host	Port	Service	Detail	Start Time	Process ID	CPU	Memory	Used Memory (MB)	Peak Used Memory (MB)
[green square]	pald00527382a	30010	compileserver		Oct 10, 2014 11:19:38 AM	12208	[CPU chart]	[Memory chart]	894	894
[green square]	pald00527382a	30000	daemon		Oct 10, 2014 11:19:19 AM	4908	[CPU chart]	[Memory chart]	0	0
[green square]	pald00527382a	30003	indexserver	master	Oct 15, 2014 11:05:53 AM	14528	[CPU chart]	[Memory chart]	5,372	5,744
[green square]	pald00527382a	30001	nameserver	master	Oct 10, 2014 11:19:21 AM	11372	[CPU chart]	[Memory chart]	1,519	1,519
[green square]	pald00527382a	30002	preprocessor		Oct 10, 2014 11:19:38 AM	12564	[CPU chart]	[Memory chart]	901	901
[green square]	pald00527382a	30005	statisticsserver	master	Oct 10, 2014 11:19:41 AM	12108	[CPU chart]	[Memory chart]	1,669	2,528
[blue square]	pald00527382a	30006	webdispatcher		Oct 10, 2014 11:20:20 AM	13408	[CPU chart]	[Memory chart]	1,276	1,276
[green square]	pald00527382a	30007	xsengine		Oct 10, 2014 11:19:41 AM	11012	[CPU chart]	[Memory chart]	1,723	1,747

# HANA XS SPS9: Local Web Dispatcher

## Full Integration into HANA Process Management

- Configuration



The screenshot shows the SAP HANA Process Management configuration interface. The top navigation bar includes tabs for Overview, Landscape, Alerts, Performance, Volumes, Configuration, System Information, and Diagnosis. The Configuration tab is currently selected. Below the tabs is a filter input field with a clear button. The main area is a table with three columns: Name, Default, and System. The table lists various configuration files and their contents. The 'webdispatcher.ini' file is expanded to show its sub-sections and values.

Name	Default	System
property_maxdb.ini		—
property_mss.ini		—
property_ntz.ini		—
property_orcl.ini		—
rsutil.ini		—
scriptserver.ini		—
statisticsserver.ini		◆
webdispatcher.ini		◆
communication		
default_read_timeout	-1	
default_read_timeout_overrule	yes	
listenport	3\$(SAPSYSTEM)06	
maxchannels	4000	
maxendpoints	4000	
profile		
trace		
xsengine.ini		◆

# HANA XS SPS9: Local Web Dispatcher

## Full Integration into HANA Process Management

- Administration
  - URL: /sap/hana/xs/wdisp/admin/public/

The screenshot shows the SAP Web Administration Interface with the title "SAP Web Dispatcher Monitor". The interface includes a sidebar menu and a main content area.

**Menu:**

- Core System
  - Monitor
  - Active Services
  - Core Thread Sta...
  - Active Connecti...
  - Trace
  - Parameters
  - Hostname Buffer
  - Release Informa...
  - Statistic
  - MPI Status
  - ICM Security Log
- SSL and Trust Conf...
- PSE Management
- HTTP Handler
  - Access Log
  - Server Cache
  - Access Handler
  - Admin Handler
  - Modification Han...
- Dispatching Module
  - SSL End To End...
  - URL Filter
  - Parameters
  - Session Dispatch...

**SAP Web Dispatcher Monitor:**

Status:	running	000		
Trace level:	1			
Process Id:	13408			
Elapsed Time / CPU Time:	123:45:39 / 00:00:43			
Created Threads:	10	13	500	20
Connections used:	1	26	2000	15724
Queue entries used:	0	4	6000	59315

No.	Thread ID	No. of Requests	Status	Request Type
0	13440	6156	idle	NOP
1	11604	6111	idle	NOP
2	2096	6000	idle	NOP
3	14176	5939	idle	NOP
4	10172	5923	idle	NOP
5	11352	5905	idle	NOP
6	8524	5891	idle	NOP
7	10684	5879	idle	NOP
8	12556	5870	running	WRITE_RESPONSE
9	13260	5872	idle	NOP



# XS Admin Tool

# HANA XS SPS9: HANA XS Admin Tool

---

**Much improved usability**

**User Self Service**

**UI of Logon Screen is Customizable**

# HANA XS SPS9: HANA XS Admin Tool – new visual design

The screenshot displays the SAP HANA XS Admin Tool interface. On the left, a sidebar menu includes 'XS Artifact Administration', 'SAML Service Provider', 'SAML Identity Provider', 'Trust Manager', 'SMTP Configurations', and 'XS Job Dashboard'. The main content area shows 'Application Objects' under 'playground sp9' with a list of packages: cds, dataGen, multiPart, restApi, smtp, xml, xsds, xsjshdb, xsunit, and zip. To the right, 'Runtime Configuration Details' are shown for the package. The 'Security & Authentication' tab is active, listing the following settings:

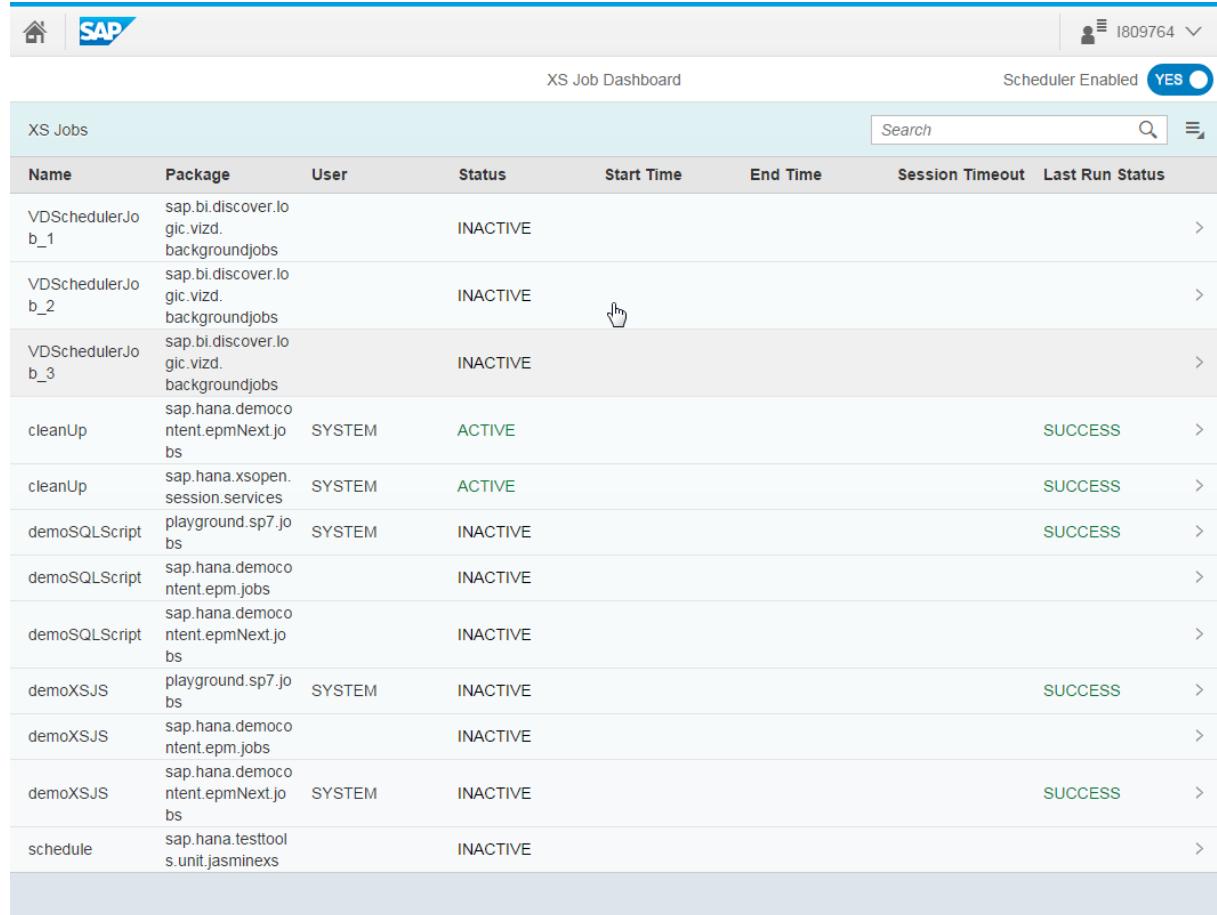
- Authentication Type: Authentication Required [Inherited]
- Connection Security: SSL Not Enforced [Inherited]
- Public Access for Sub-Packages: Allowed [Inherited]
- Authentication Methods:
  - SAML (disabled)
  - SPNEGO (disabled)
  - X509 (enabled)
  - SAP Logon/Assertion Ticket (disabled)
  - Form Based (enabled)
  - Basic (enabled)

At the bottom right of the main area are 'Edit', 'Reset', and a refresh icon.

# HANA XS SPS9: HANA XS Admin Tool: Scheduled Jobs – New Management UI

## New Management UI (XS Job Dashboard)

URL: /sap/hana/xs/admin/jobs/



The screenshot shows the SAP XS Job Dashboard. At the top right, there is a user icon and the ID '1809764'. To the right of the user icon, it says 'Scheduler Enabled' with a blue 'YES' button. Below the header is a search bar with the placeholder 'Search' and a magnifying glass icon. The main area is titled 'XS Jobs' and contains a table with the following columns: Name, Package, User, Status, Start Time, End Time, Session Timeout, and Last Run Status. There are 13 rows of data in the table.

Name	Package	User	Status	Start Time	End Time	Session Timeout	Last Run Status
VDSchedulerJob_1	sap.bi.discover.lo gic.vizd. backgroundjobs		INACTIVE				
VDSchedulerJob_2	sap.bi.discover.lo gic.vizd. backgroundjobs		INACTIVE				
VDSchedulerJob_3	sap.bi.discover.lo gic.vizd. backgroundjobs		INACTIVE				
cleanUp	sap.hana.democontent.epmNextJobs	SYSTEM	ACTIVE			SUCCESS	
cleanUp	sap.hana.xsopen.session.services	SYSTEM	ACTIVE			SUCCESS	
demoSQLScript	playground.sp7.jobs	SYSTEM	INACTIVE			SUCCESS	
demoSQLScript	sap.hana.democontent.epm.jobs		INACTIVE				
demoSQLScript	sap.hana.democontent.epmNextJobs		INACTIVE				
demoXSJS	playground.sp7.jobs	SYSTEM	INACTIVE			SUCCESS	
demoXSJS	sap.hana.democontent.epm.jobs		INACTIVE				
demoXSJS	sap.hana.democontent.epmNextJobs	SYSTEM	INACTIVE			SUCCESS	
schedule	sap.hana.testtools.unit.jasmine		INACTIVE				

# HANA XS SPS9: HANA XS Admin Tool: Scheduled Jobs – New Management UI Continued

## New Management UI (XS Job Dashboard)

Drill into details of a single job definition from Job Dashboard or from the XS Admin tool

The screenshot shows the SAP HANA XS Admin Tool interface. At the top, there's a header with the SAP logo and a user ID (I809764). Below the header, the page title is "XS Job Details". There are two tabs: "Job Details" (which is selected) and "Configuration". Under the "Job Details" tab, the job information is displayed:

Name:	cleanUp
Description:	Clean Up Old Cookies
Action:	sap.hana.democontent.epmNext.procedures::clean_up_old_cookies

Below this, the "Schedules" section is shown as a table:

ID	XSCron	Description	Parameter	Planned...	Status	Start...	Finish...	Time Taken (ms)
56	*****2 0 0	Will run once a day at 2:00am		2014-10-15...	SCHEDULED			
56	*****2 0 0	Will run once a day at 2:00am		2014-10-14...	SUCCESS	2014-10-14...	2014-10-14...	0
56	*****2 0 0	Will run once a day at 2:00am		2014-10-13...	SUCCESS	2014-10-13...	2014-10-13...	0
56	*****2 0 0	Will run once a day at 2:00am		2014-10-12...	SUCCESS	2014-10-12...	2014-10-12...	0
56	*****2 0 0	Will run once a day at 2:00am		2014-10-11...	SUCCESS	2014-10-11...	2014-10-11...	0
56	*****2 0 0	Will run once a day at 2:00am		2014-10-10...	SUCCESS	2014-10-10...	2014-10-10...	0
56	*****2 0 0	Will run once a day at 2:00am		2014-10-09...	SUCCESS	2014-10-09...	2014-10-09...	0
56	*****2 0 0	Will run once a day at 2:00am		2014-10-08...	SUCCESS	2014-10-08...	2014-10-08...	0
56	*****2 0 0	Will run once a day at 2:00am		2014-10-07...	SUCCESS	2014-10-07...	2014-10-07...	0

At the bottom of the schedule table, there are buttons for "Add Schedule" and "Delete Schedule".

# HANA XS SPS9: Customize Login Screen

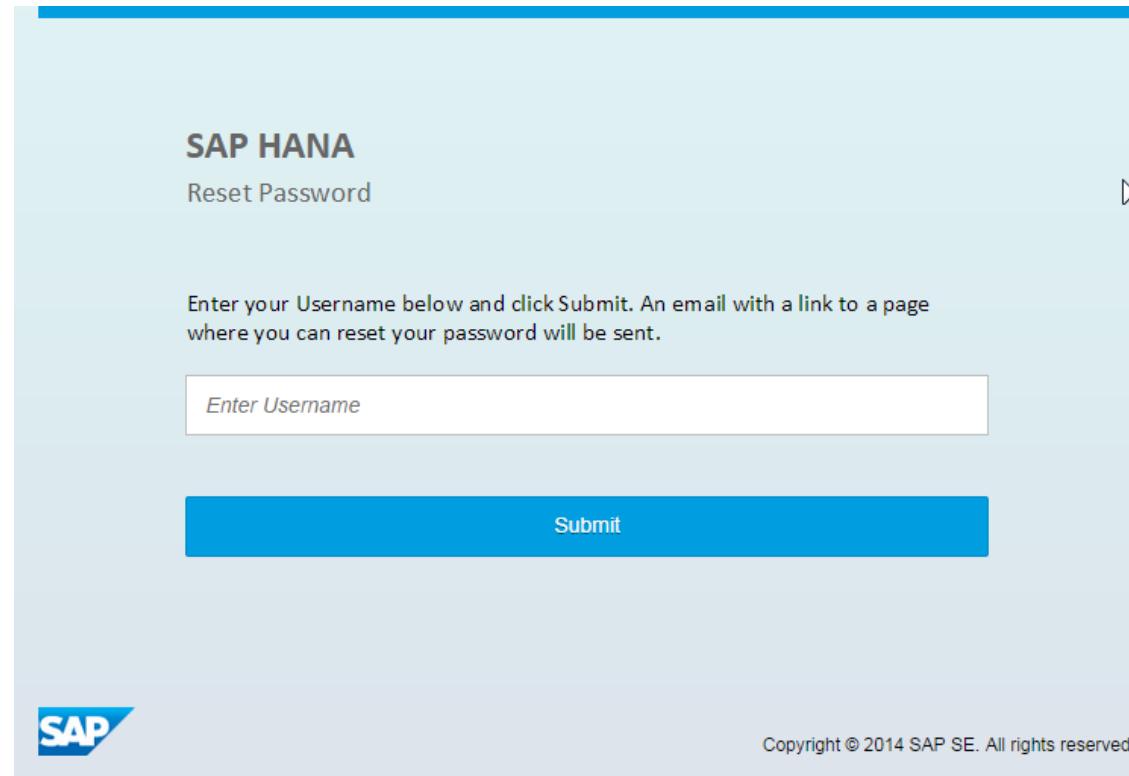
**Custom background image set via xsengine.ini -> httpserver parameter**

xsengine.ini		◆
application_container		◆
communication		◆
debugger		◆
geocoding		◆
httpserver		◆
developer_mode		● true
embedded		● true
listenport	3\$(SAPSYSTEM)08	
login_screen_background_image		● /sap/ui5/1/Background1.png

# HANA XS SPS9: User Self Service

---

**/sap/hana/xs/selfService/user/resetPassword.html**



# HANA XS SPS9: User Self Service

---

**/sap/hana/xs/selfService/user/requestAccount.html**

The screenshot shows a web page titled "SAP HANA" with the sub-section "Request Account". The page has a light blue header and a white content area. It contains two input fields: "Enter Username" and "Enter Email Address", both with placeholder text. Below these fields is a large blue "Submit" button. At the bottom left is the SAP logo, and at the bottom right is the copyright notice "Copyright © 2014 SAP SE. All rights reserved."

SAP HANA

Request Account

Enter Username

Enter Email Address

Submit

SAP

Copyright © 2014 SAP SE. All rights reserved.

# HANA XS SPS9: User Self Service – Admin

**/sap/hana/xs/selfService/admin/**

The screenshot shows the SAP HANA XS SPS9 User Self Service - Admin interface. The left sidebar lists categories: User Requests (0 items), Domains (0 items), Email Addresses (0 items), and IP Ranges (0 items). The 'Domains' category has a 'Blacklisted' and 'Whitelisted' count of 0, with a cursor icon pointing to the 'Whitelisted' link. The right panel is titled 'User Requests' and 'User Self Service Requests'. It features a 'Users' tab with a download icon, a description 'List of User self service requests', and a search/filter bar with columns: Username, Request attempts, Request origin, Email address, Administration, Request Status, and Deactivation. Below the search bar, it says 'No data'. At the bottom, there are buttons for 'Add to blacklist' and 'Activate and Notify'.

# HANA XS SPS9: User Self Service – User Profile

**/sap/hana/xs/formLogin/profile/**

The screenshot shows the SAP HANA XS SPS9 User Self Service - User Profile interface. At the top, there is a navigation bar with a home icon, the SAP logo, and a user ID (1809764) with a dropdown arrow. Below the navigation bar, the title "Manage Profile" is displayed. The main content area is divided into two sections: "Change Password" on the left and "Preferences" on the right. The "Change Password" section contains three input fields: "Old Password" (placeholder: Enter Old Password), "New Password" (placeholder: Enter New Password), and "Repeat Passw..." (placeholder: Repeat Password). The "Preferences" section contains three dropdown menus: "Date Format" (YYYY-MM-DD), "Time Format" (HH24:MI), and "Locale" (English (en)). A "Save" button is located at the bottom right of the form.

Change Password		Preferences	
Old Password:	Enter Old Password	Date Format:	YYYY-MM-DD
New Password:	Enter New Password	Time Format:	HH24:MI
Repeat Passw...	Repeat Password	Locale:	English (en)

Save



# HANA Test Tools

# HANA XS SPS9: Test Framework

---

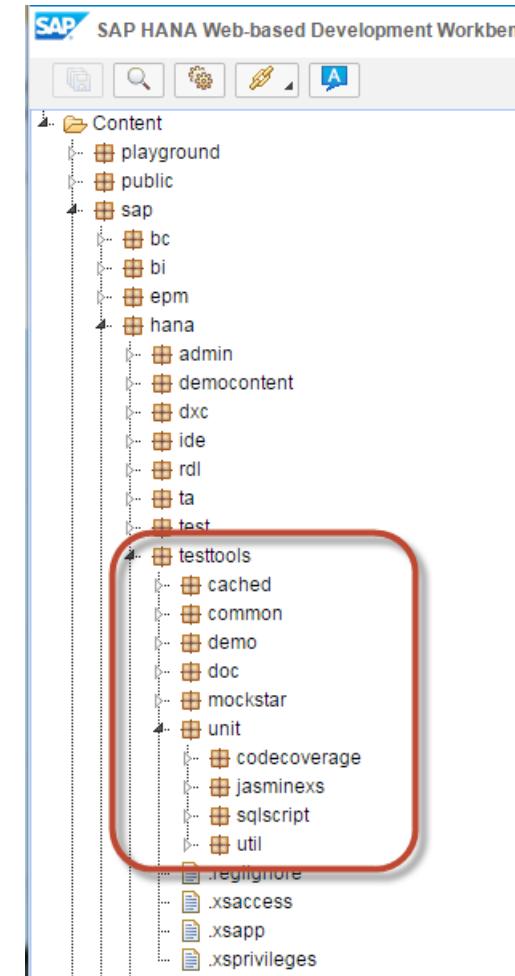
**Unit Test Framework**

**Mock Framework**

**(shipped with HANA, but not pre-installed)**

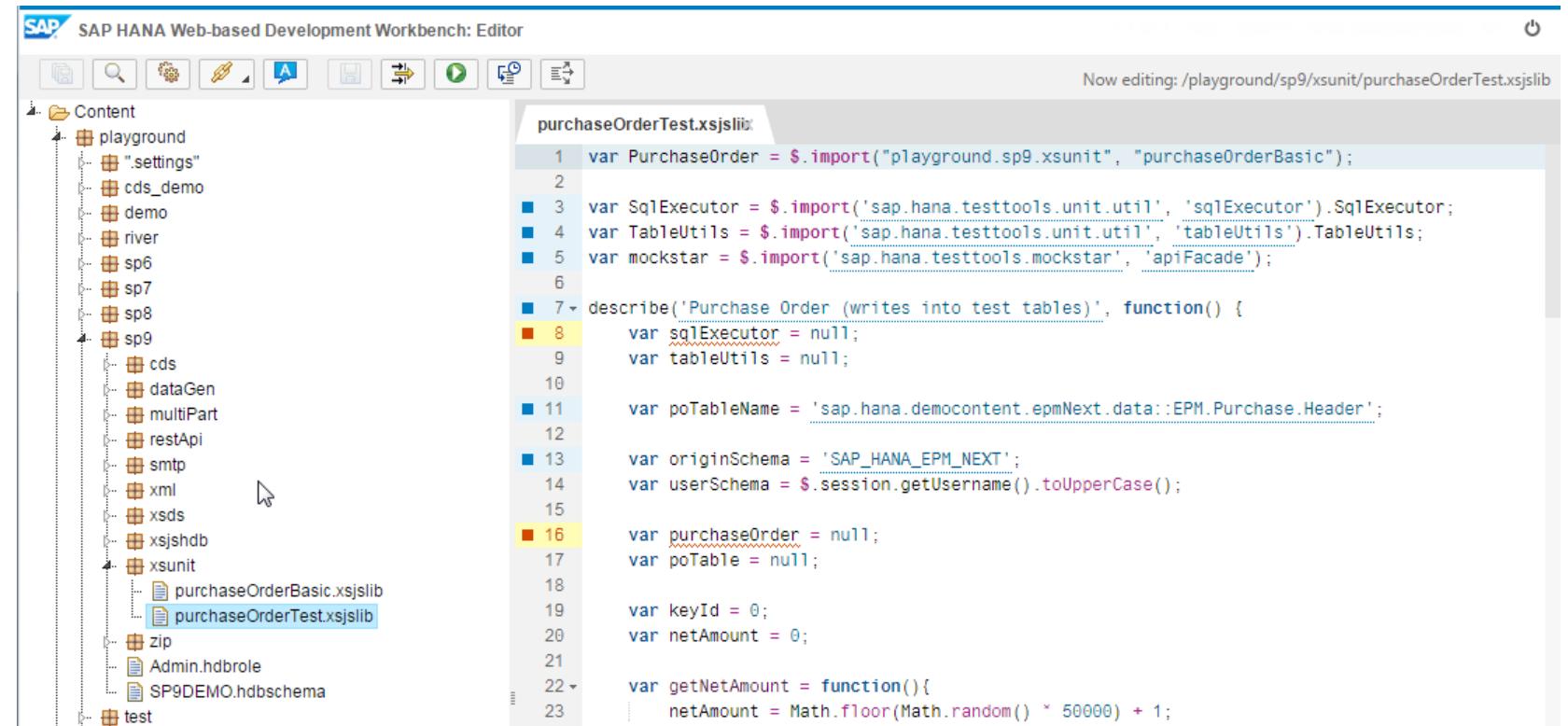
# Contents of HANA\_TEST\_TOOLS Delivery Unit

- XS Unit Test Framework and Test Runner (unit.jsaminexs)
- XSUnit Test Coverage for XS JavaScript (unit.codecoverage)
- Test Isolation Framework (mockstar)
- Test Helper (unit.util)
- Developer Documentation (doc)
- Demos (demo)



# How to create an XSUnit Test

- One common tool for testing XS JavaScript, database content: SQLScript, Views
- Based on open-source library, extended by custom assertions, test utilities
- Test code and test data are developed beside your productive code, in the same HANA instance
- Tests are implemented in XS JavaScript in HANA Studio or the Web-based Development Workbench

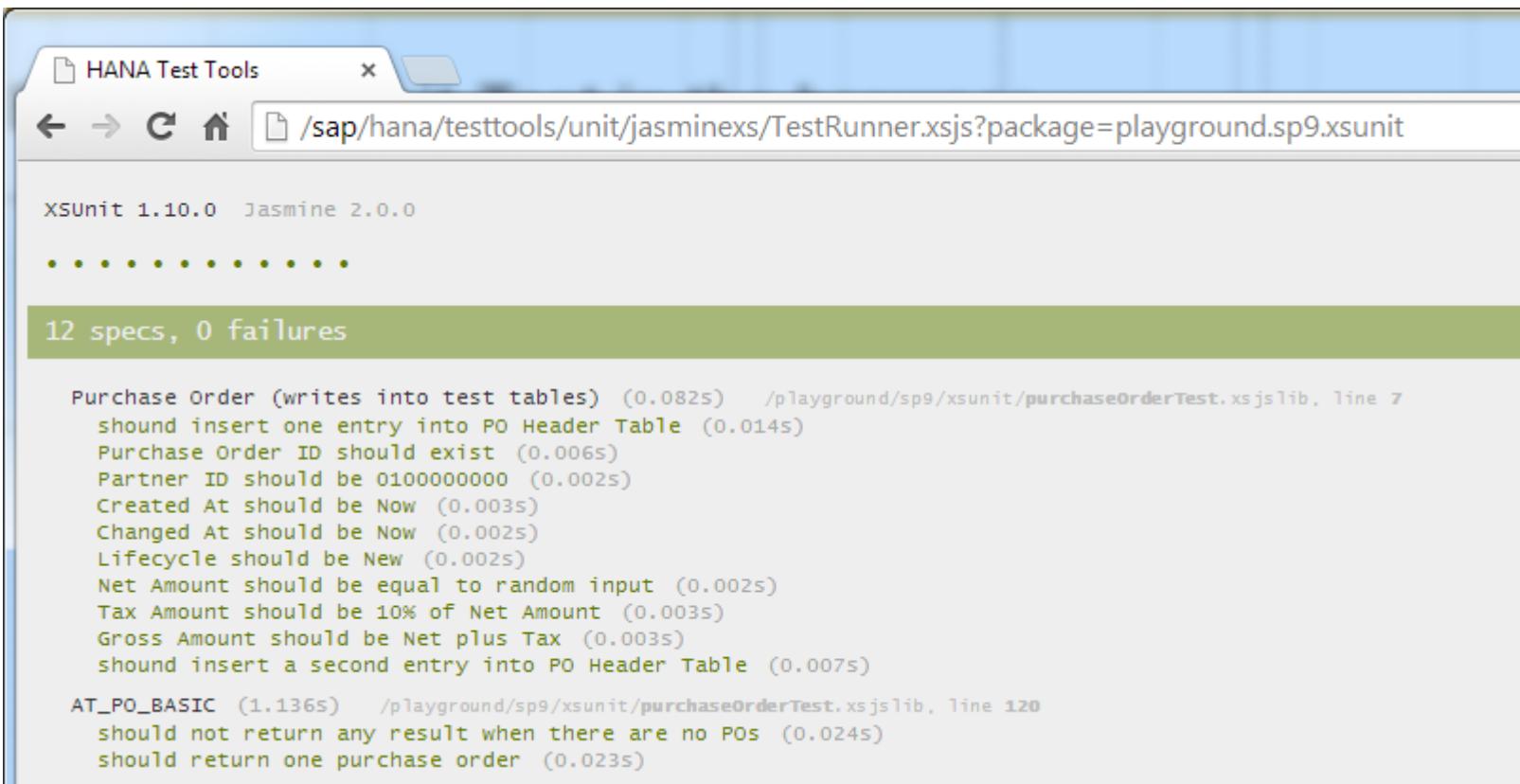


The screenshot shows the SAP HANA Web-based Development Workbench Editor. The left sidebar displays a file tree under 'Content' with several projects like playground, demo, river, sp6, sp7, sp8, and sp9, each containing various sub-folders such as cds, dataGen, multiPart, restApi, smtp, xml, xsds, xsjshdb, and xsunit. Under the xsunit folder of the sp9 project, there are two files: purchaseOrderBasic.xsjslib and purchaseOrderTest.xsjslib, with the latter being the currently edited file. The right pane shows the code editor with the file 'purchaseOrderTest.xsjslib'. The code is written in XS JavaScript and includes imports for PurchaseOrder, SqlExecutor, TableUtils, and mockstar from sap.hana.testtools. It defines a describe block for 'Purchase Order (writes into test tables)' and initializes variables for sqlExecutor, tableUtils, poTableName, originSchema, userSchema, purchaseOrder, and poTable. It also defines a getNetAmount function that generates a random net amount between 0 and 50,000.

```
1 var PurchaseOrder = $.import("playground.sp9.xsunit", "purchaseOrderBasic");
2
3 var SqlExecutor = $.import('sap.hana.testtools.util', 'sqlExecutor').SqlExecutor;
4 var TableUtils = $.import('sap.hana.testtools.util.util', 'tableUtils').TableUtils;
5 var mockstar = $.import('sap.hana.testtools.mockstar', 'apiFacade');
6
7 describe('Purchase Order (writes into test tables)', function() {
8     var sqlExecutor = null;
9     var tableUtils = null;
10
11     var poTableName = 'sap.hana.democontent.epmNext.data::EPM.Purchase.Header';
12
13     var originSchema = 'SAP_HANA_EPM_NEXT';
14     var userSchema = $.session.getUsername().toUpperCase();
15
16     var purchaseOrder = null;
17     var poTable = null;
18
19     var keyId = 0;
20     var netAmount = 0;
21
22     var getNetAmount = function(){
23         netAmount = Math.floor(Math.random() * 50000) + 1;
```

# Execute XSUnit Test in the browser

- URL:  
</sap/hana/testtools/unit/jasminexs/TestRunner.xsjs?package=<>>
- package = the repository package your tests are located



XSUnit 1.10.0 Jasmine 2.0.0

• • • • • • • •

12 specs, 0 failures

```
Purchase Order (writes into test tables) (0.082s) /playground/sp9/xsunit/purchaseOrderTest.xsjslib, line 7
    should insert one entry into PO Header Table (0.014s)
    Purchase Order ID should exist (0.006s)
    Partner ID should be 0100000000 (0.002s)
    Created At should be Now (0.003s)
    Changed At should be Now (0.002s)
    Lifecycle should be New (0.002s)
    Net Amount should be equal to random input (0.002s)
    Tax Amount should be 10% of Net Amount (0.003s)
    Gross Amount should be Net plus Tax (0.003s)
    should insert a second entry into PO Header Table (0.007s)

AT_PO_BASIC (1.136s) /playground/sp9/xsunit/purchaseOrderTest.xsjslib, line 120
    should not return any result when there are no POs (0.024s)
    should return one purchase order (0.023s)
```

# Execute XSUnit Test in the browser: URL Parameters

Parameter	Required	Description	Example
tags	no	Comma-separated list of tags to restrict the tests to be executed.	tags=integration,long_running
reporter	no	Complete path to module providing an implementation of the Jasmine reporter interface. With this parameter a custom reporter can be passed to publish the test results in an application specific format	reporter=sap.hana.testtools.unit.jasminexs.reporter.db.dbReporter has the same effect like format=db
profile	no	Name of a "profile" defined in the test which filters the tests to be executed on the basis of tags.	profile=end2end

# Execute XSUnit Test in the browser: URL Parameters (Continued)

Parameter	Required	Description	Example
pattern	no	Naming pattern that identifies the .xsjslib files that contain the tests. If not specified, the pattern “*Test” is applied. You can use wildcards “?” and “*” to match a single and multiple arbitrary characters respectively.	pattern=Test* would match all xsjslib files beginning with “Test”; In these files you may then manually import other test files.
package	yes	Package that acts as starting point for discovering the tests. If not otherwise specified by parameter “pattern” all .xsjslib files in this package and its sub-packages conforming to the naming pattern “*Test” will be assumed to contain tests and will be executed.	package=sap.hana.te sttools.demo

# Execute XSUnit Test in the browser: URL Parameters (Continued)

---

Parameter	Required	Description	Example
format	no	Specifies the output format the test runner uses to report test results. By default, the results will be reported as HTML document. This parameter has no effect if a custom reporter is provided via parameter “reporter”.	format=html shows results in HTML; format=junit or format=xml outputs XML conforming to schema used by Jenkins; format=json outputs results in JSON format; format=db writes test results to database tables

# Execute XSUnit from SAP Web-based Development Workbench

The screenshot illustrates the SAP HANA Web-based Development Workbench interface, specifically focusing on the execution of XSUnit tests.

**Left Panel (File Explorer):** Shows the project structure under the 'xsunit' folder. The 'purchaseOrderTest.xsjslib' file is selected. A context menu is open, with the 'Invoke Tests' option highlighted.

**Middle Panel (Editor):** Displays the code editor for the 'purchaseOrderTest.xsjslib' file. The code implements unit tests for purchase orders using Jasmine and SQLExecutor. Several test cases are highlighted in yellow, indicating they have been run.

**Right Panel (XS Unit Test Runner):** Shows the results of the executed tests. It includes a summary of tracked code coverage and a detailed list of test results for the 'Purchase Order' scenario, including assertions for Purchase Order ID, Partner ID, Created At, Lifecycle, Net Amount, Tax Amount, Gross Amount, and more.

# Analyze Test Coverage

The screenshot shows a web browser window with the URL [/sap/hana/testtools/unit/jasminexs/GetTestResults.xsjs?runid=2&coverage=1](http://pal00527382a:8000/sap/hana/testtools/unit/jasminexs/GetTestResults.xsjs?runid=2&coverage=1). The page displays the results for a test named "playground.sp9\_xsunit.purchaseOrderBasic". The coverage statistics are shown in the first row:

Blanket.js results	Covered/Total Smts.	Coverage (%)
1. playground.sp9_xsunit.purchaseOrderBasic	62/94	65.96 %

The code listing below the table shows a portion of a JavaScript function. Line 28, which contains the statement `value = rs.getString(i);`, is highlighted with a red background, indicating it was not covered by the test.

```
1  /**
2  * @author i809764
3  */
4  var conn = $.db.getConnection();
5  var pstmt;
6  var rs;
7  var query;
8
9 /**
10 * @function base "class" with reusable functions
11 */
12 function base() {
13 }
14
15 /**
16 * @function takes a Record Set and returns the value based upon the metadata type
17 * @param {int} i - counter for the current position in the record set
18 * @param {object} rs - Record Set object
19 * @param {object} meta - meta data object
20 * @returns {*} the value for the current position in the record set
21 */
22 base.prototype.getValueFromRS = function(i, rs, meta) {
23     var value;
24     switch (meta.getColumnType(i)) {
25         case $.db.types.VARCHAR:
26         case $.db.types.CHAR:
27             value = rs.getString(i);
28             break;
29         case $.db.types.NVARCHAR:
30         case $.db.types.NCHAR:
31         case $.db.types.SHORTTEXT:
32             value = rs.getNString(i);
33             break;
34     }
35 }
```

# View XSUnit Test Results

HANA XS TEST [Jenkins] - Microsoft Internet Explorer  
http://vmw4916.wdf.sap.corp:8080/jenkins/view/HANA%20XS%20TEST/

File Edit View Favorites Tools Help  
Favorites Unit Testing on HANA... XS Unit Test Framework... HANA XS TEST [Je...  
Find: identifier Previous Next Options search D048418 | log out  
ENABLE AUTO REFRESH

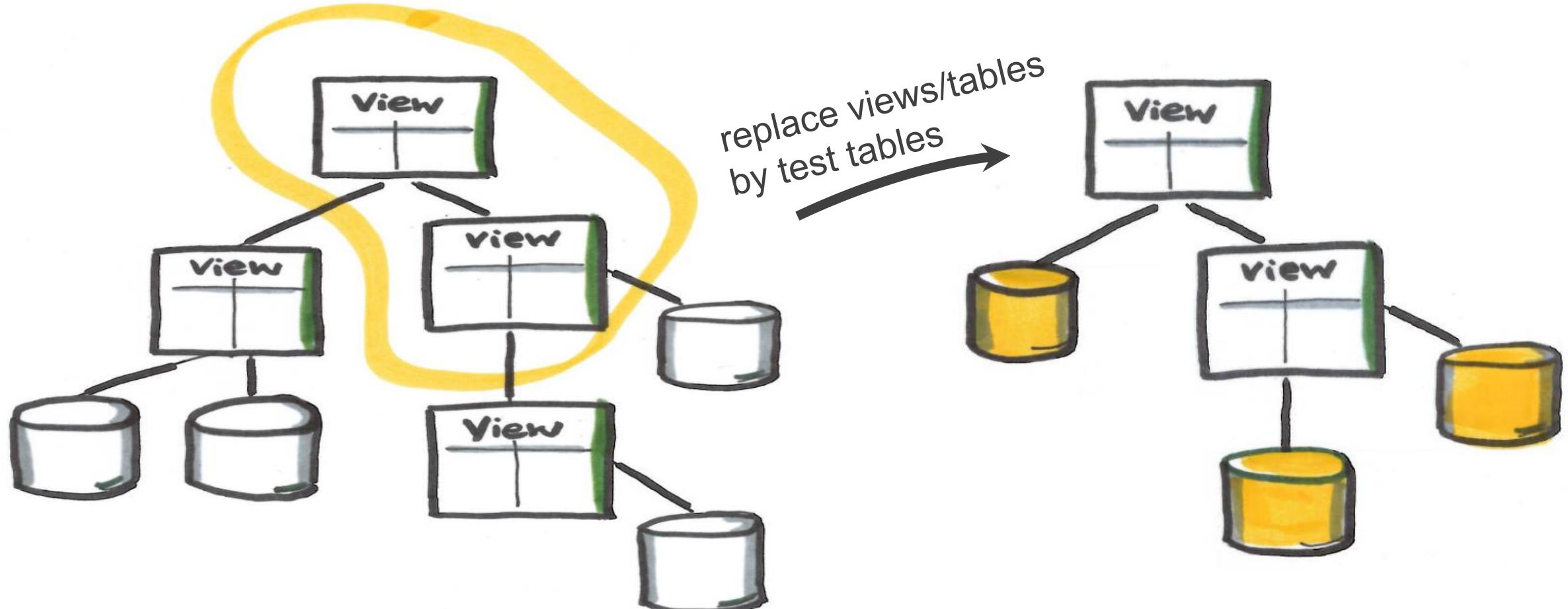
## Jenkins

New Job People Build History Edit View Delete View Project Relationship Check File Fingerprint My Views

Welcome to CPO I2M AgileSE Product Team Jenkins system

'SE Frameworks and Plugins			All	HANA XS TEST	Jenkins TWB Integration Demo Jobs		Quality on Submit	RSR_View	+
S	W	Name			Last Success	Last Failure	Last Duration	Changelist number	Line Coverage
		<a href="#">HANA XS TEST DEMO</a>			2 hr 32 min (#75)	N/A	19 sec		
		<a href="#">HANA XS TEST JASMINE TESTS</a>			2 hr 13 min (#84)	N/A	1,3 sec		
		<a href="#">HANA XS TEST JUNIT TESTS</a>			26 days (#21)	9 days 5 hr (#24)	44 sec		
		<a href="#">HANA XS TEST MOCKSTAR INTEGRATION TESTS</a>			2 hr 12 min (#115)	N/A	1 sec		
		<a href="#">HANA XS TEST MOCKSTAR UNIT TESTS</a>			2 hr 51 min (#100)	N/A	1,3 sec		
		<a href="#">HANA XS TEST TEST UTILITIES ABAP</a>			2 hr 35 min (#118)	N/A	0,54 sec		

# Mocking Framework





# CDS



# HANA XS SPS9: CDS

---

## Mission Statement

- CDS provides an enriched data model
- CDS allows for graceful life-cycle management
- CDS allows for *extending* the meta-model

## Features coming with SP9

- Multi-File
- Table Lifecycle
- Associations
- DCL (Instance filtering)
- Much improved editing support in Studio and Web-based Development Workbench

# CDS new features

---

- Support for view complex features such as sub-selects
- Multi-file support
- Calc-view support
- Unmanaged Associations
- Annotation (enums, array, etc)
- Lifecycle management support for data migration
- GIS-Types
- Core HANA data types
- Currency Conversion
- Enterprise search via full text index

## Lifecycle Management Example

---

**SQL:**

```
CREATE TABLE PARTNER(NAME char(30), TYPE char(1));
```

**Adding new field:**

```
ALTER TABLE PARTNER ADD(GROSS int);
```

## Lifecycle Management Example

CDS:

```
entity PARTNER {  
    NAME: String(30);  
    TYPE: String(1);  
    GROSS: Integer;  
};
```

Simply add the field. The  
necessary SQL is generated

# CDS

## Lifecycle Management Example

CDS:

```
entity PARTNER {  
    NAME: String(30);  
    TYPE: String(1);  
    GROSS: Integer Integer64;  
};
```

Upon activation data is  
migrated automatically

This Lifecycle Management supports a wide variety of scenarios

- Conversion from same named HDBTABLE
- Change of key / data type
- Removal of columns

# Multi-file: File 1

---

```
namespace playground.sp9.cds;
@Schema: 'SP9DEMO'
context ContextA {
    type T1 : Integer;
    context ContextAI {
        type T2 : String(20);
        type T3 {
            a : Integer;
            b : String(88);
        };
    };
};
```

## Multi-file: File 2

```
namespace playground.sp9.cds;
using playground.sp9.cds::ContextA.T1;
using playground.sp9.cds::ContextA.ContextAI as ic;
using playground.sp9.cds::ContextA.ContextAI.T3 as ict3;
@Schema: 'SP9DEMO'
context ContextB {
    type T10 {
        a : T1;          // Integer
        b : ic.T2;       // String(20)
        c : ic.T3;       // structured
        d : type of ic.T3.b; // String(88)
        e : ict3;        // structured
    };
    context ContextBI {
        type T1 : String(7); // hides the T1 coming from the first using declaration
        type T2 : T1;        // String(7)
    };};
```

# CDS

## Enumerations

---

```
namespace playground.sp9.cds;
@Schema: 'SP9DEMO'
context enumerations {

    type Color : String(10) enum { red = 'FF0000'; g = '00FF00'; b = '0000FF'; };

    annotation MyAnnotation {
        a : Integer;
        b : String(20);
        c : Color;
        d : Boolean;
    };};

};};
```

# CDS

## New Types

---

```
entity SomeTypes {  
    a : hana.ALPHANUM(10);  
    b : hana.SMALLINT;  
    c : hana.TINYINT;  
    d : hana.SMALLDECIMAL;  
    e : hana.REAL;  
    h : hana.VARCHAR(10);  
    i : hana.CLOB;  
    j : hana.BINARY(10);  
    k : hana.ST_POINT;  
    l : hana.ST_GEOGRAPHY; };
```

# CDS

## Views – OrderBy

---

```
view EmployeesView as select from Employee
{
    orgUnit,
    salary }
order by salary desc;
```

# CDS

## Views – Case

---

```
entity MyEntity {  
    key id : Integer;  
    a : Integer;  
    color : String(1); };  
view MyView as select from MyEntity {  
    id,  
    case color  
        when 'R' then 'red'  
        when 'G' then 'green'  
        when 'B' then 'blue'  
        else 'black'  
    end as color,  
    case when a < 10 then 'small'  
        when 10 <= a and a < 100 then 'medium'  
        else 'large'  
    end as size };
```

# CDS

## Unmanaged Associations

---

```
entity Employee {  
    key id : String(256);  
    officId : Integer; };  
  
entity Room {  
    key id : Integer;  
    inhabitants : Association[*] to Employee on inhabitants.officId = id; };  
  
entity Thing {  
    key id : Integer;  
    parentId : Integer;  
    parent : Association[1] to Thing on parent.id = parentId;  
    children : Association[*] to Thing on children.parentId = id; };
```

# Backlink workaround via Unmanaged Associations

---

```
entity Header {  
    key id : Integer;  
    items : Association[*] to Item on items.headerId = id;  
    description : String(120);  
};
```

```
entity Item {  
    key headerId : Integer;  
    key id : Integer;  
    header : Association[1] to Header on header.id = headerId;  
    description : String(120);  
};
```

# Many to Many relationships

---

```
entity Employee {  
    key id : Integer;  
    name : String(80);  
    projectLinks : Association[*] to E2P on projectLinks.e_id = id; };  
  
entity Project {  
    key id : Integer;  
    name : String(80);  
    employeeLinks : Association[*] to E2P on employeeLinks.p_id = id; };  
  
entity E2P {  
    key e_id : Integer;  
    key p_id : Integer;  
    projects : Association[*] to Project on projects.id = p_id;  
    employees : Association[*] to Employee on employees.id = e_id; };
```

# Many to Many relationships (continued)

---

```
view EmployeesWithProjects as select from Employee {  
    name as EmployeeName,  
    projectLinks.projects.id as projectId,  
    projectLinks.projects.name as projectName };
```

```
view ProjectsWithEmployees as select from Project {  
    name as projectName,  
    employeeLinks.employees.id as EmployeeId,  
    employeeLinks.employees.name as EmployeeName };
```

# Full Text Index Annotation

```
entity Header {  
    key id : Integer;  
    @SearchIndex.text: { enabled: true }  
    @SearchIndex.fuzzy: { enabled: true }  
    description : String(120);  
};
```

The screenshot shows the SAP Studio interface with the following details:

- Project:** backlink.hdbdd
- Table:** ORG - SP9DEMO.playground.sp9.cds::backlink.Header
- Table Name:** playground.sp9.cds::backlink.Header
- Schema:** SP9DEMO
- Indexes Tab:** The "Indexes" tab is selected.
- Index Details:** A single index entry is listed:

Name	Type	Indexed Columns
_ESH_FULLTEXT_820201...	Full-Text Index	"description"



# Extended Programming Model



# Extended Programming Model

## XSDS

# Goals of seamless HDB consumption in XSJS

---

## General HANA consumption

- ✓ Native JavaScript embedding of HANA database artifacts
- ✓ Structured API calls instead of low-level SQL interface (ODBC)
- ✓ Less boilerplate coding, e.g., result set conversion

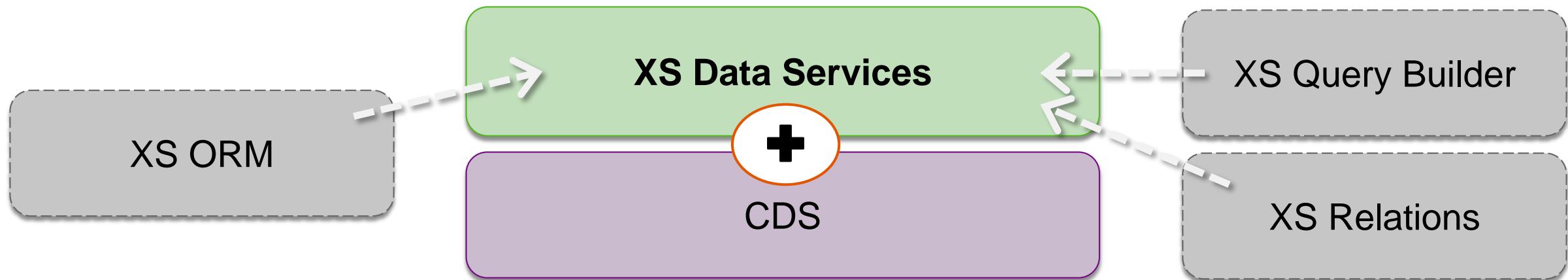
## XSDS: Native CDS consumption

- ✓ Import CDS entities as native JavaScript objects
- ✓ Understand CDS metadata for working with JavaScript objects

# XS Data Services (XSDS)

**XSDS as Native CDS Client and Query Builder for XSJS**

Succeeds three HANA Consumption projects by Platform Research





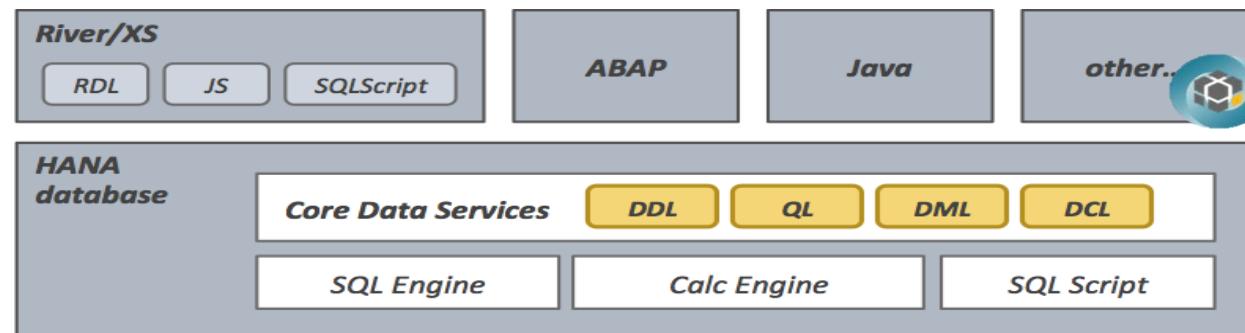
# **Extended Programming Model**

## **XSDS – Data Models**

# Core Data Services

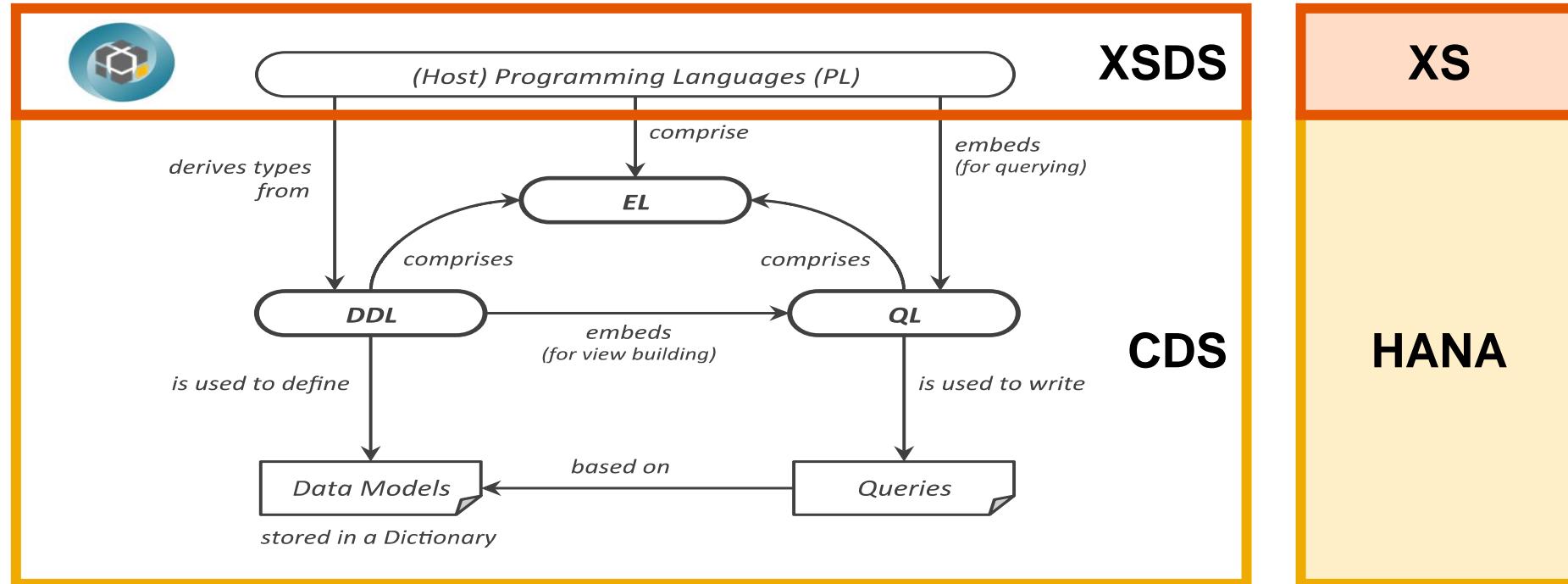
Core Data Services (CDS) are a cross-platform set of concepts and tools to define ***semantically rich data models*** for SAP HANA applications. A central part of CDS is the ***object-relational mapping*** that links semantically meaningful ***entities*** to their technical representation as ***records*** in the HANA database.

**CDS is central HANA functionality provided via SQL interface**



# CDS and XSDS

**XSDS is the native CDS embedding for JavaScript in the XS Engine**



# Sample CDS data model

```
namespace sap.hana.xsopen.dbutils.xsds.examples.data;
context bboard {
    entity user {
        key uid: Integer not null;
        Name: String(63) not null;
        Email: String(63);
        Created: UTCDateTime;
    };
    type text {
        Text: String(255);
        Lang: String(2);
    };
    entity post {
        key pid: Integer not null;
        Title: String(63) not null;
        Text: text;
        Author: association [1] to user;
        Parent: association [0..1] to post;
        Rating: Integer;
        // Comments: association [0..*] to comment via backlink Parent;
        // Tags: association [0..*] to tag via entity tags_by_post;
        Created: UTCDateTime;
    };
}
```

# Consume CDS model data in XS

## Import and extend CDS entity definitions

```
var User = XSDS.$importEntity("demo.bboard", "bboard.user");
```

Reads available metadata on types, keys, associations

Supports extension, projection, renaming of entity definitions

Already supports CDS *via backlink*, *via entity* associations

```
var Post = XSDS.$importEntity("demo.bboard", "bboard.post", {  
    Comments: {  
        $association: { $entity: Comment, $viaBacklink: "Post" }  
    }  
});
```

# Import pre-generation

---

## CDS import costly operation

XS Engine has no session state

→ (Re-)Import of CDS metadata for every single requests

```
var User = XSDS.$importEntity("demo.bboard", "bboard.user");
```

## Pregeneration of imports improves performance

Serialize result of imports into single library file

```
var UserLib = $.import("demo.bboard", "User");
var User = UserLib.demo.bboard.User.entity;
```

# Main XSDS features

## Entity Manager

Lightweight ORM

Navigation to associations

Data consistency

Limited query functionality



## Query Builder

Ad-hoc queries

Based on CDS QL (WIP)

Manual consistency

Full HANA support

## XS Data Services

Entity instances and unmanaged values

## CDS

Entities and Types

## HANA



# Extended Programming Model

## XSDS – Managed Mode

# XSDS managed entities

---

## Create, retrieve, update, and delete instances of entities

```
var post = Post.$get({ pid: 101 });
```

Result is plain JavaScript object

```
if (post.Author.Name === "Alice") post.Rating++;
```

Optional lazy retrieval of associated instances to avoid "n+1 problem"

Optimized JOIN strategy to minimize DB queries (WIP)

Write changes to database explicitly

```
post.$save();
```

Manager tracks local changes for minimal updates

# Reference: instance manipulation

## Retrieve by key

```
var post = Post.$get({ pid: 101 });
```

## Update instance

```
post.Author.Name = "Alice";  
post.$save();
```

## Create new instance

```
var user = new User({ Name: "Alice", Created: new Date() });  
user.$save();
```

## Delete instance

```
post.$discard();
```

# Instance management

---

## Entity instances are managed

Instances are singleton objects with "personality"

Associated instances are stored by reference

## Upside: Runtime guarantees on consistency

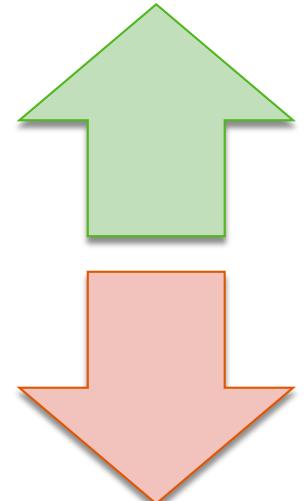
Single consistent view on every instance (enforced by \$persist)

Automatic association management (1:1, 1:n, m:n)

## Downside: Entity cache

Additional overhead for instance management

Only limited subset of database functionality for instance retrieval



# Consistent instance view

XSJS

```
{ pid: 101, Title: "First", Author: }
```

```
{ pid: 103, Title: "Third", Author: }
```

```
{ uid: 2, Email: "bob@sap.ag" }
```

```
{ uid: 1, Email: "alice@sap.se" }
```

HANA

pid	Title	Author.uid
101	First	1
103	Third	1

uid	Email
1	alice@sap.ag
2	bob@sap.ag

# Database and transaction handling

---

## **Transparent database and transaction handling**

Minimal XSDS runtime (metadata processing, entity cache, type conversion)

Exposes/reuses DB functionality as much as possible (e.g., constraints checks)

## **XSDS uses single DB connection and single transaction**

Suited for single-threaded, per-request execution model of XS Engine

Supports auto commit, explicit commits, and rollbacks

```
post1.$save();
post2.$save();
XSDS.Transaction.$commit();
```

Supports connection configurations (.xssqlcc)



# Extended Programming Model

## XSDS – Unmanaged Mode

# CDS queries from XSJS: motivation

„Get the names of authors which commented on their own posts and the text of that comment“

## CDS Metadata Definition

```
entity post {  
    key pid: Integer not null;  
    Author: association [1] to user;  
};  
  
entity user {  
    key uid: Integer notnull;  
    Name: String(63) not null;  
};  
  
entity comment {  
    key cid: Integer not null;  
    Text: text;  
    Author: association[1] to user;  
    Post: association[1] to post; //via backlink  
};
```

## XS Code

```
var rs = conn.prepareStatement('  
SELECT a."Name", c."Text.Text"  
FROM "bboard.post" p  
JOIN "bboard.user" a ON p."Author.uid"=a."uid"  
JOIN "bboard.comment" c ON p."pid"=c."Post.pid"  
JOIN "bboard.user" ca ON c."Author.uid"=ca."uid"  
WHERE a."uid"=ca."uid").execute();  
while (rs.next()) {  
    rs.getInt(1); rs.getString(2);  
}
```

### Redundant Code:

JOINS could be derived  
from metadata

### Leaking Abstraction:

Column name can only be  
derived from table definition

### Convenience:

„String-based“ query  
interface  
& manual result set  
processing

# XSDS queries

---

## Ad-hoc queries

Construct general queries in a JavaScript-like way

```
var query = Post.$query().$where(Post.Author.Name.$eq("Alice"));
var results = query.$execute();
```

Yields plain JavaScript object as unmanaged value, not instance

## Query Language

Incremental query building

Operators \$project, \$where, \$matching, ...

Full support for CDS path navigation

Expression language based on CDS Query Language using fluent API

Alternative, more concise JSON-like selection notation for restricted subset

# Demo: XSDS queries - \$project

## Projections with CDS navigation expressions

```
var res = Post.$query().$project({ Author: {Name: true},  
                                  Comments: {Author: {Name: true}}},  
                               "Title:  
"TitleOfPost")  
    .$execute();
```

Result:

```
[{ Author: {  
            Name: "Alice"  
        },  
        Comments: {  
            Author: {  
                Name: "Bob"  
            }  
        },  
        TitleOfPost: "First Post!"  
    }, ...]
```

# Demo: XSDS queries - \$where

## Complex where conditions:

```
var selfCommentingPosts =  
Post.$query().$where(Post.Comments.Author.pid.$eq(Post.Author.pid))  
.$execute();
```

## Predefined operators of expression language:

\$eq, \$ne, \$gt, \$lt, \$ge, \$le, \$like, \$unlike

## Escape to any SQL operator:

```
var somePost = Post.$query()  
.$where(Post.Rating.$prefixOp('SQRT').$gt(2)).$execute();
```

translated to:

WHERE SQRT ("t0"."Rating") > 2

# Demo: XSDS queries - \$matching

## Selection using “template” for the result

```
var interestingPosts = Post.$query().$matching({  
    Rating: {  
        $gt: 2  
    },  
    Tags: {  
        Name: "+1"  
    }  
}).execute();
```



Unmanaged version of \$find, \$findAll, uses same template language

# Demo: incremental query construction

## Incremental construction with immutable query objects

```
// build query without DB call
var qBadPosts = Post.$query().$where(Post.Author.pid)
    .$eq(Post.Comments.Author.pid));
// refine query
var qStarBadPosts = qBadPosts.$where(Post.Rating.$gt(4));
```

## Explicit trigger

```
// trigger actual DB calls
var badTitles = qBadPosts.$project(
    { Title: true, Author: {Name : true}}).$execute();
var recentBadTitles = qStarBadPosts.$project({ Title: true })
    .$execute();
```

# Reference: JSON expressions

---

## Simple expressions

```
Post.$find({ mandt: "001", pid: 101 });
```

## Complex expressions

```
Post.$find({ Title: { $like: "First%" },
             Rating: { $gt: 1, $lt: 3 },
             Created: { $lt: Date.now() - 3600 } });
```

## JSON Expression Language Operators

```
$eq $ne $lt $le $gt $ge $like $unlike $null
```

# Query builder

---

## Optimized query construction

Allows for incremental query construction without database roundtrips

Immutable query objects support sharing of subqueries

Just-in-time translation into plain SQL, using needed joins

```
SELECT "t0.Author"."Name" AS "t0.Author.Name",
       "t0.Author"."Email" AS "t0.Author.Email",
       "t0"."Title" AS "t0.Title"
  FROM "sap.hana.xsopen.xsds.examples::bboard.post" "t0"
 JOIN "sap.hana.xsopen.xsds.examples::bboard.user" "t0.Author"
    ON "t0"."Author.uid"="t0.Author.uid"
 WHERE ("t0"."Rating" > 2) AND ("t0.Author"."Name" = 'Alice')
```



# **Extended Programming Model**

## **XSDS - Procedures**

# XS Procedures Library

---

A standard reusable library  
shipped by SAP to simplify  
Stored Procedure calls from  
XSJS and make they “feel”  
like JavaScript functions

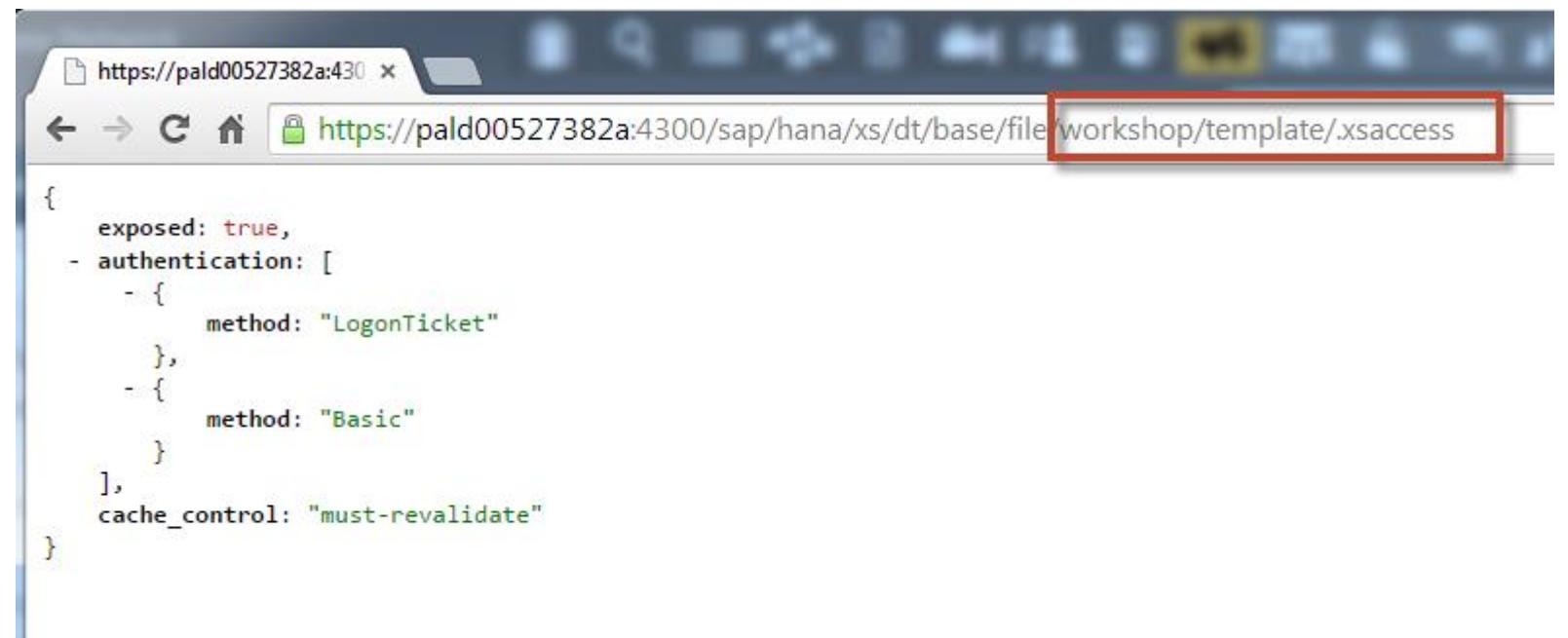
```
*****  
// Call procedure using the xsProcedure API  
*****  
var getBpAddressesByRole = XSProc.procedureOfSchema("SAP_HANA_EPM_NEXT",  
    "sap.hana.democontent.epmNext.procedures::get_bp_addresses_by_role");  
var results = getBpAddressesByRole( { IM_PARTNERROLE: partnerRole}, conn );  
var jsonOut = results; //You can reference the output parameter directly as well ... results.EX_BP_ADDRESSES  
conn.close();
```



# Extended Programming Model Repository REST API

# SAP HANA REST API

- An implementation of the Orion Server API within HANA
- Designed to allow development tools access to HANA capabilities, especially the Repository.



A screenshot of a web browser window displaying a JSON configuration file for a REST API endpoint. The URL in the address bar is `https://pald00527382a:4300/sap/hana/xs/dt/base/file/workshop/template/.xsaccess`. The JSON content shows authentication methods and cache control settings:

```
{  
  exposed: true,  
  authentication: [  
    - {  
        method: "LogonTicket"  
      },  
    - {  
        method: "Basic"  
      }  
  ],  
  cache_control: "must-revalidate"  
}
```

# SAP HANA REST API – server API parts

---

- File API
- Workspace API
- Transfer API
- Metadata API
- Change Tracking API
- OData Services
- Info API

# SAP HANA REST API – File API

---

- **File API**
- Workspace API
  - Enables you to browse and manipulate files and directories via HTTP
- Transfer API
- Metadata API
  - Basic HTTP methods GET, PUT, and POST to send requests
- Change Tracking API
  - JSON is used as the default representation format
- OData Services
  - Can also retrieve metadata and previous versions of the content
- Info API

# SAP HANA REST API – Workspace API

---

- File API
- **Workspace API**
  - Enables you to create and manipulate workspaces and projects via HTTP
  - Workspace and project concepts taken over from Eclipse
  - Both ORION concepts are technically mapped to the SAP HANA XS package concept
  - When you create a project, it is an SAP HANA XS sub package in the specified workspace package
- Transfer API
- Metadata API
- Change Tracking API
- OData Services
- Info API

# SAP HANA REST API – Transfer API

---

- File API
- Workspace API
- **Transfer API**
  - Is used to import and export packages and files
  - Resumable, chunked uploads of file content
  - Export not yet implemented. Use the File API for export.
- Metadata API
- Change Tracking API
- OData Services
- Info API

# SAP HANA REST API – Metadata API

---

- File API
- Workspace API
- Transfer API
- **Metadata API**
  - Provides services to support **search** and **auto completion** scenarios
  - it is possible to retrieve metadata for tables, views, procedures, functions, sequences, and schemas as well as CDS metadata
- Change Tracking API
- OData Services
- Info API

# SAP HANA REST API – Change Tracking API

---

- File API
- Workspace API
- Transfer API
- Metadata API
- **Change Tracking API**
  - Enables you to make use of specific lifecycle-management features included with the SAP HANA Repository via HTTP
  - For example: you can ensure that an export operation includes only the latest *approved* versions of repository objects
- OData Services
- Info API

# SAP HANA REST API – OData Services

---

- File API
- Workspace API
- Transfer API
- Metadata API
- Change Tracking API
- **OData Services**
  - A set of OData services that enable access to configuration data stored in the SAP HANA environment
  - Includes details on the following:
    - Delivery Units
    - INI Configuration files
    - M\_FEATURES
- Info API

# SAP HANA REST API – Info API

---

- File API
- Workspace API
- Transfer API
- Metadata API
- Change Tracking API
- OData Services
- **Info API**
  - Can be used to display information about the current version of the REST API.
  - Includes a description of the current version of the delivery unit and the number of commands (API entry points) that are currently supported by the REST API

# SAP HANA REST API - SapBackPack

---

- SAP specific functionality (like activation) added via the additional parameter - SapBackPack

```
oSapBackPack.Workspace='SHINE_DATA';
oSapBackPack.Activate = true;
sapBackPack = JSON.stringify(oSapBackPack);

$.ajax({
    url : oUrl,
    type : 'PUT',
    data : frames.response,
    headers : {
        "Content-Type" : "text/plain; charset=UTF-8",
        "SapBackPack" : sapBackPack,
        "X-CSRF-Token" : frames.securityToken
    },
    success : function(response) {
        if (response === 'Success') {
            frames.close();
        }
    }
});
```



# SAP River

# SAP River – What's New in SPS 09?

---

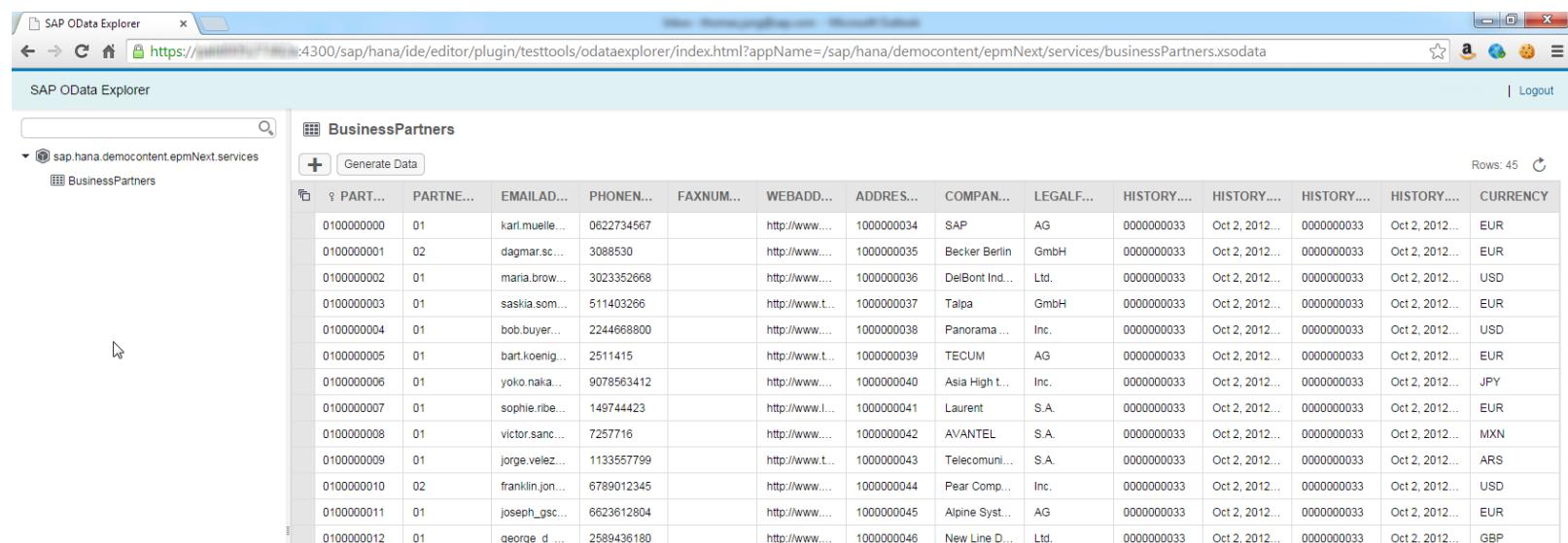
**The benefits of the SAP River language (RDL) application development environment were validated by partners and early adaptors. However, based on feedback we received, and consistent with our strategic direction of building open and standard environments, SAP has decided to abandon a proprietary language approach, and to reapply and integrate the SAP River assets and principles within a cloud based development environment as part of the HANA Cloud Platform.**

**The SAP River language will therefore no longer be available as a stand-alone development environment in SAP HANA.**

# SAP River assets reused

**SAP River application explorer  
rebuilt as SAP OData Explorer  
in SPS09**

General OData test and data  
generation tool which supports  
**XSO DATA** services



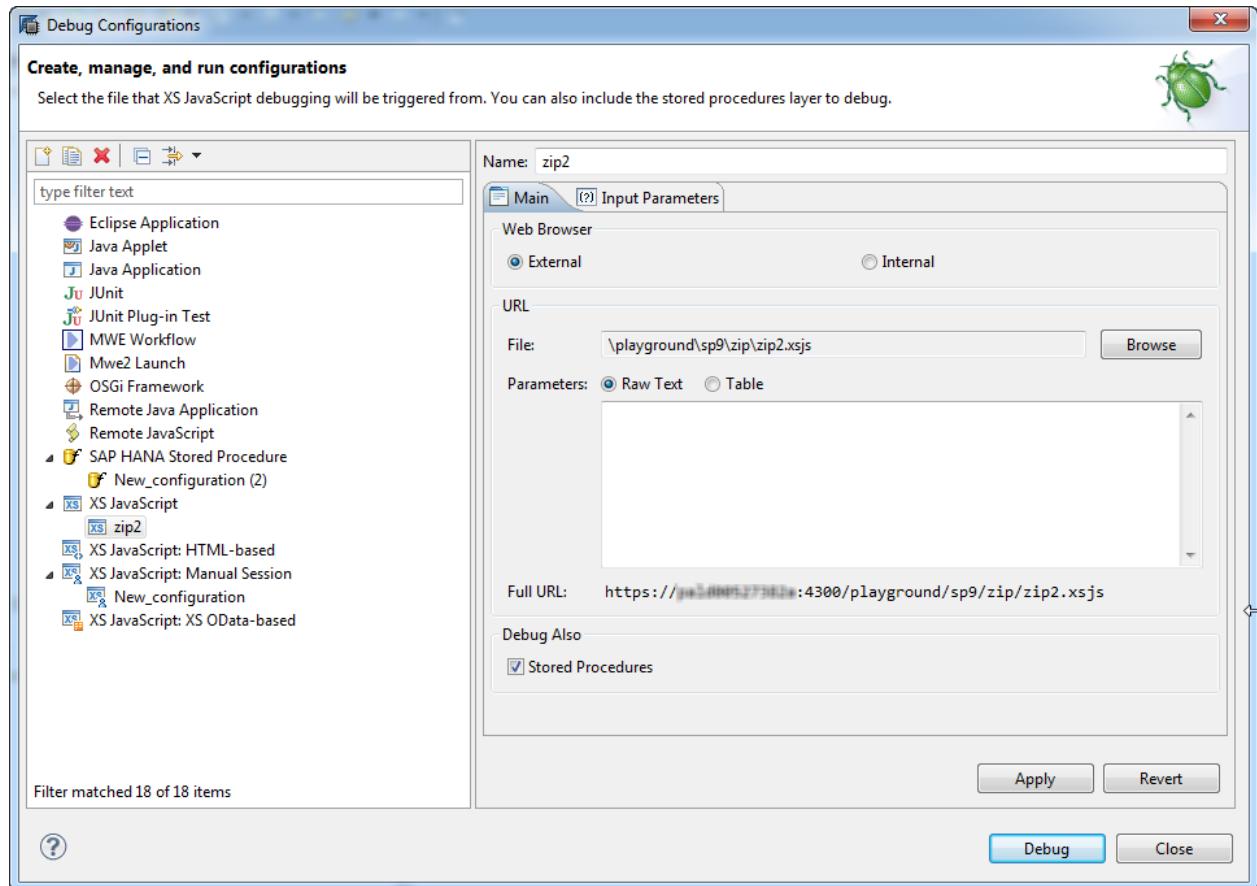
The screenshot shows a browser window titled "SAP OData Explorer" displaying data from an XSODATA service. The URL in the address bar is `https://...:4300/sap/hana/ide/editor/plugin/testtools/odataexplorer/index.html?appName=/sap/hana/democontent/epmNext/services/businessPartners.xsodata`. The page title is "BusinessPartners". A search bar and a "Generate Data" button are visible. The main content is a table with 45 rows of data, each representing a business partner. The columns are labeled: PART..., PARTNE..., EMAILAD..., PHONEN..., FAXNUM..., WEBADD..., ADDRES..., COMPAN..., LEGALF..., HISTORY..., HISTORY..., HISTORY..., HISTORY..., CURRENCY. The data includes various names, addresses, and company details.

PART...	PARTNE...	EMAILAD...	PHONEN...	FAXNUM...	WEBADD...	ADDRES...	COMPAN...	LEGALF...	HISTORY...	HISTORY...	HISTORY...	HISTORY...	CURRENCY
0100000000	01	karl.muelle...	0622734567		http://www....	1000000034	SAP	AG	0000000033	Oct 2, 2012...	0000000033	Oct 2, 2012...	EUR
0100000001	02	dagmar.sc...	3088530		http://www....	1000000035	Becker Berlin	GmbH	0000000033	Oct 2, 2012...	0000000033	Oct 2, 2012...	EUR
0100000002	01	maria.brow...	3023352668		http://www....	1000000036	DeiBont Ind...	Ltd.	0000000033	Oct 2, 2012...	0000000033	Oct 2, 2012...	USD
0100000003	01	saskia.som...	511403266		http://www.t...	1000000037	Talpa	GmbH	0000000033	Oct 2, 2012...	0000000033	Oct 2, 2012...	EUR
0100000004	01	bob.buyer...	2244668800		http://www....	1000000038	Panorama ...	Inc.	0000000033	Oct 2, 2012...	0000000033	Oct 2, 2012...	USD
0100000005	01	bart.koenig...	2511415		http://www.t...	1000000039	TECUM	AG	0000000033	Oct 2, 2012...	0000000033	Oct 2, 2012...	EUR
0100000006	01	yoko.naka...	9078563412		http://www....	1000000040	Asia High ...	Inc.	0000000033	Oct 2, 2012...	0000000033	Oct 2, 2012...	JPY
0100000007	01	sophie.ribe...	149744423		http://www.l...	1000000041	Laurent	S.A.	0000000033	Oct 2, 2012...	0000000033	Oct 2, 2012...	EUR
0100000008	01	victor.sanc...	7257716		http://www....	1000000042	AVANTEL	S.A.	0000000033	Oct 2, 2012...	0000000033	Oct 2, 2012...	MXN
0100000009	01	jorge.velez...	1133557799		http://www.t...	1000000043	Telecomuni...	S.A.	0000000033	Oct 2, 2012...	0000000033	Oct 2, 2012...	ARS
0100000010	02	franklin.jon...	6789012345		http://www....	1000000044	Pear Comp...	Inc.	0000000033	Oct 2, 2012...	0000000033	Oct 2, 2012...	USD
0100000011	01	joseph_gst...	6623612804		http://www....	1000000045	Alpine Syst...	AG	0000000033	Oct 2, 2012...	0000000033	Oct 2, 2012...	EUR
0100000012	01	george_d...	2589436180		http://www....	1000000046	New Line D...	Ltd.	0000000033	Oct 2, 2012...	0000000033	Oct 2, 2012...	GBP

# SAP River concepts realized in alternative ways

## Integrated one-click debugging

Step from the application server layer logic directly in database layer logic within one debug session



# SAP River concepts realized in alternative ways - XSDS

## XSDS: Native CDS consumption

- ✓ Import CDS entities as native JavaScript objects
- ✓ Understand CDS metadata for working with JavaScript objects

## Import and extend CDS entity definitions

```
var User = XSDS.$importEntity("demo.bboard",  
    "bboard.user");
```

Reads available metadata on types, keys, associations

Supports extension, projection, renaming of entity definitions

Already supports CDS *via backlink*, *via entity* associations

```
var Post = XSDS.$importEntity("demo.bboard", "bboard.post", {  
    Comments: {  
        $association: { $entity: Comment, $viaBacklink: "Post" }}});
```

# SAP River concepts realized in alternative ways – XSDS (continued)

## Entity Manager

Lightweight ORM

Navigation to associations

Data consistency

Limited query functionality



## Query Builder

Ad-hoc queries

Based on CDS QL (WIP)

Manual consistency

Full HANA support

## XS Data Services

Entity instances and unmanaged values

## CDS

Entities and Types

## HANA

# Disclaimer

---

**This presentation outlines our general product direction and should not be relied on in making a purchase decision. This presentation is not subject to your license agreement or any other agreement with SAP.**

**SAP has no obligation to pursue any course of business outlined in this presentation or to develop or release any functionality mentioned in this presentation. This presentation and SAP's strategy and possible future developments are subject to change and may be changed by SAP at any time for any reason without notice.**

**This document is provided without a warranty of any kind, either express or implied, including but not limited to, the implied warranties of merchantability, fitness for a particular purpose, or non-infringement. SAP assumes no responsibility for errors or omissions in this document, except if such damages were caused by SAP intentionally or grossly negligent.**

# How to find SAP HANA documentation on this topic?

- In addition to this learning material, you can find SAP HANA platform documentation on SAP Help Portal knowledge center at [http://help.sap.com/hana\\_platform](http://help.sap.com/hana_platform).
- The knowledge centers are structured according to the product lifecycle: installation, security, administration, development:

## SAP HANA Platform SPS

- [What's New – Release Notes](#)
- [Installation](#)
- [Administration](#)
- [Development](#)
- [References](#)

The screenshot shows the SAP Help Portal interface. At the top, there's a navigation bar with links for SAP Business Suite, Technology, Application Lifecycle Mgmt, Mobile, SAP for Industries, Analytics, SAP Best Practices, Database, SAP Business One, SAP In-Memory Computing, Cloud, and Additional Information. Below the navigation bar, the main content area is titled "SAP HANA Platform". It displays the "Support Package Stack 08 (Last Update: August 21, 2014, Revision 82)". On the left, there's a sidebar with sections for SAP In-Memory Computing, SAP HANA, SAP HANA Cloud Platform, SAP HANA One, Innovations for SAP Business Suite, Innovations for SAP NetWeaver, and Innovations for On-Demand Solutions. The main content area lists various topics under the SAP HANA section, such as What's New – Release Notes, Installation and Upgrade Information, Security Information, System Administration and Maintenance Information, Modeling Information, Development Information, Reference Information, End-User Information, and Additional Information.

- Documentation sets for SAP HANA options can be found at [http://help.sap.com/hana\\_options](http://help.sap.com/hana_options):

## SAP HANA Options

- [SAP HANA Advanced Data Processing](#)
- [SAP HANA Dynamic Tiering](#)
- [SAP HANA Enterprise Information Management](#)
- [SAP HANA Predictive](#)
- [SAP HANA Real-Time Replication](#)
- [SAP HANA Smart Data Streaming](#)
- [SAP HANA Spatial](#)

### SAP HANA Options



SAP HANA options provide additional features to the base edition of the SAP HANA platform. To use the SAP HANA options in a production system, you must purchase the appropriate software license from SAP. The SAP HANA options listed below are available in connection with the base edition of the SAP HANA platform.

SAP HANA Options
<a href="#">SAP HANA Advanced Data Processing</a>
<a href="#">SAP HANA Dynamic Tiering</a>
<a href="#">SAP HANA Enterprise Information Management</a>
<a href="#">SAP HANA Predictive</a>
<a href="#">SAP HANA Real-Time Replication</a>
<a href="#">SAP HANA Smart Data Streaming</a>
<a href="#">SAP HANA Spatial</a>



# Thank you

## Contact information

Thomas Jung  
SAP HANA Product Management  
[AskSAPHANA@sap.com](mailto:AskSAPHANA@sap.com)

# © 2014 SAP SE or an SAP affiliate company. All rights reserved.

---

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP SE or an SAP affiliate company.

SAP and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP SE (or an SAP affiliate company) in Germany and other countries. Please see <http://global12.sap.com/corporate-en/legal/copyright/index.epx> for additional trademark information and notices.

Some software products marketed by SAP SE and its distributors contain proprietary software components of other software vendors.

National product specifications may vary.

These materials are provided by SAP SE or an SAP affiliate company for informational purposes only, without representation or warranty of any kind, and SAP SE or its affiliated companies shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP SE or SAP affiliate company products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

In particular, SAP SE or its affiliated companies have no obligation to pursue any course of business outlined in this document or any related presentation, or to develop or release any functionality mentioned therein. This document, or any related presentation, and SAP SE's or its affiliated companies' strategy and possible future developments, products, and/or platform directions and functionality are all subject to change and may be changed by SAP SE or its affiliated companies at any time for any reason without notice. The information in this document is not a commitment, promise, or legal obligation to deliver any material, code, or functionality. All forward-looking statements are subject to various risks and uncertainties that could cause actual results to differ materially from expectations. Readers are cautioned not to place undue reliance on these forward-looking statements, which speak only as of their dates, and they should not be relied upon in making purchasing decisions.