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)

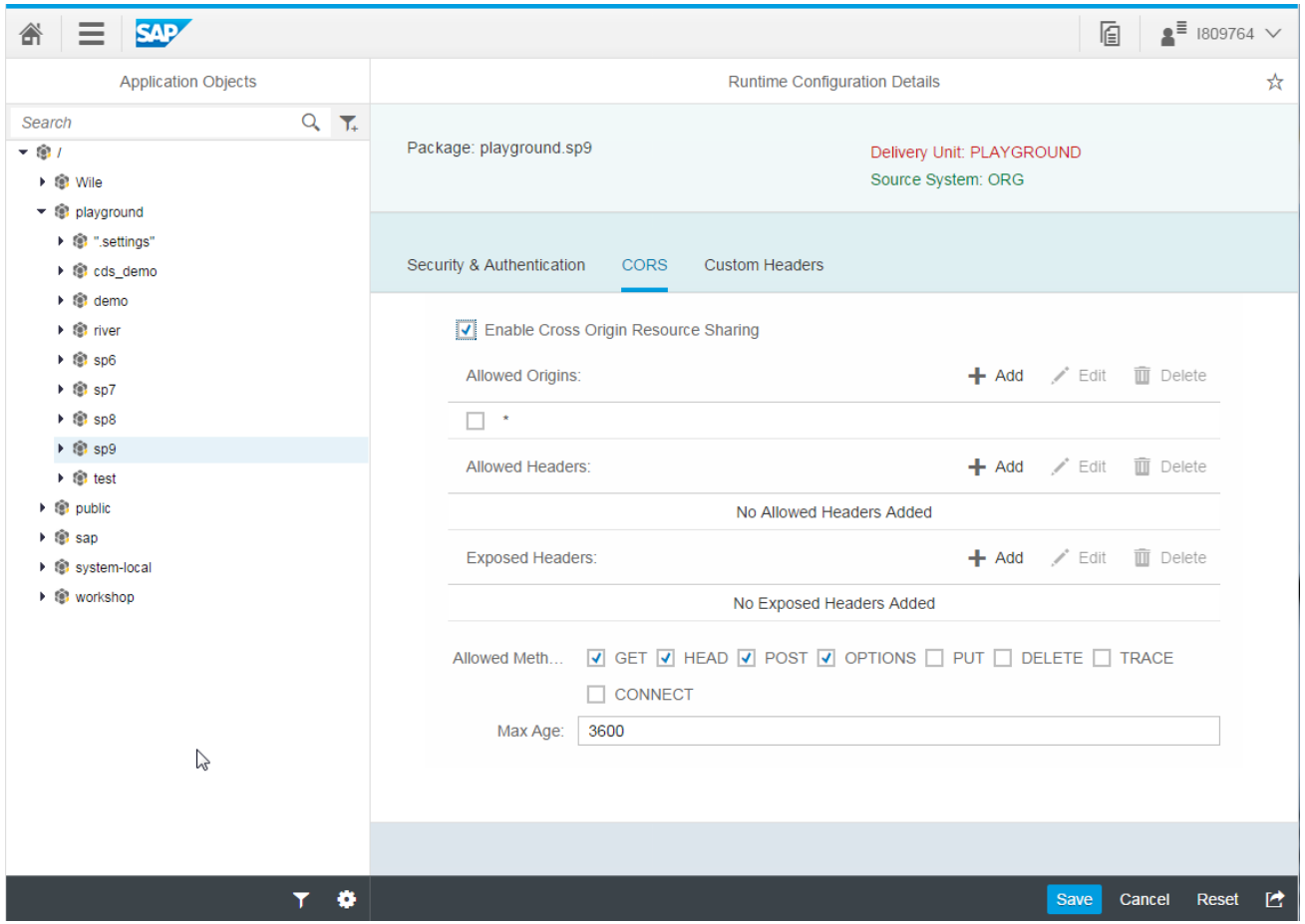
SAML Authentication in Authorization header

Support of Virus Scan Interface (VSI) for applications

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

Full CORS support (Cross-Origin Resource Sharing)

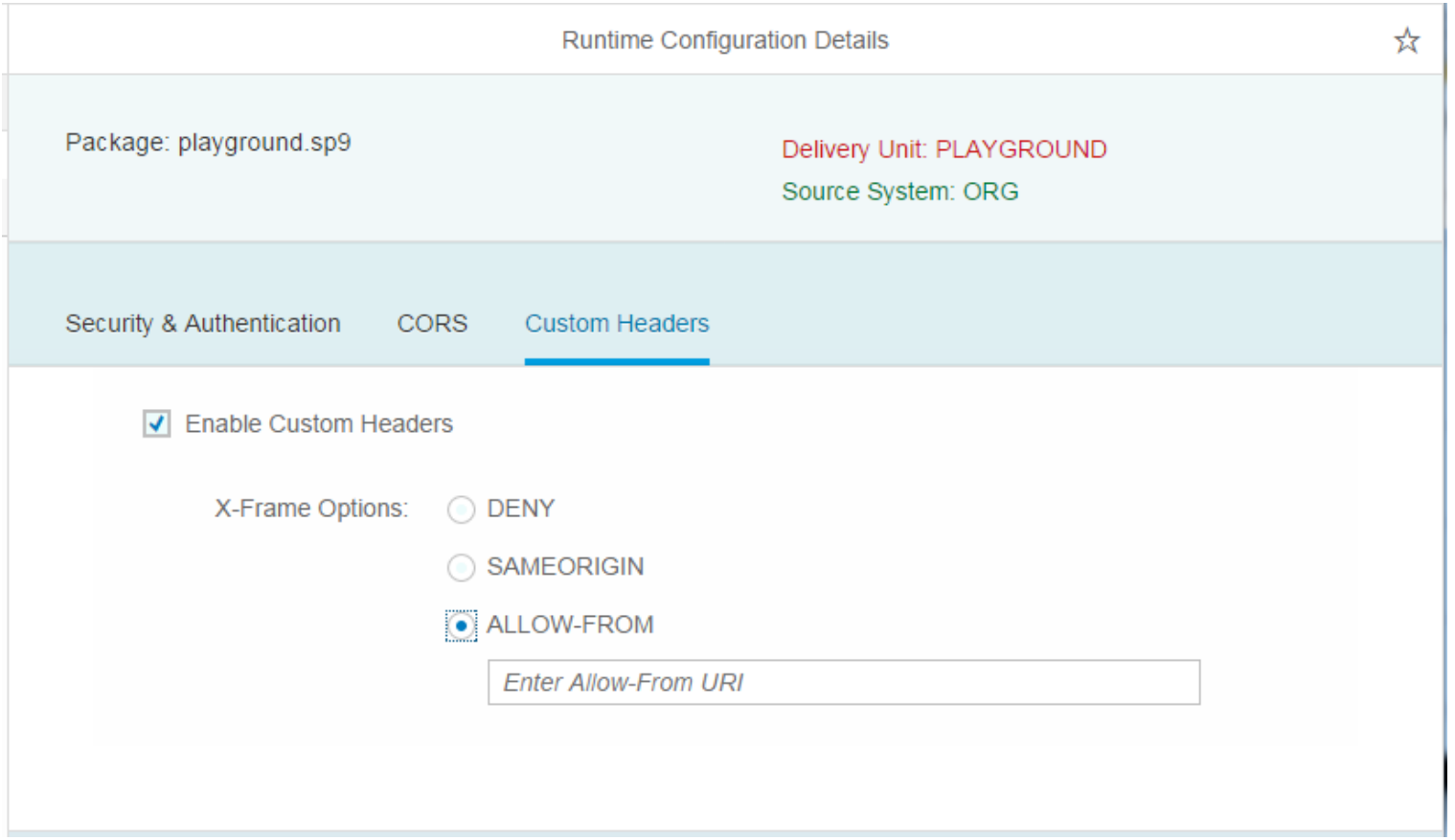
- **Filtering by origins, headers, and methods**



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**



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

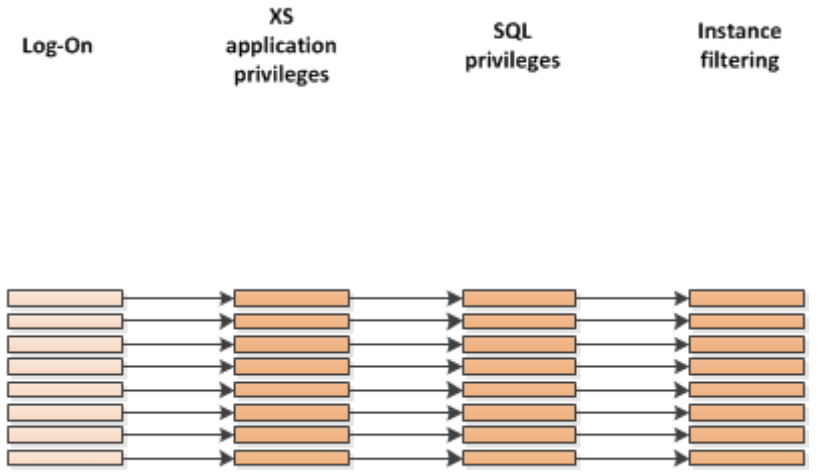
Improved Support for Close Cooperation with ABAP Applications

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

Programmatic alternative (SqlScript) to XS application privileges

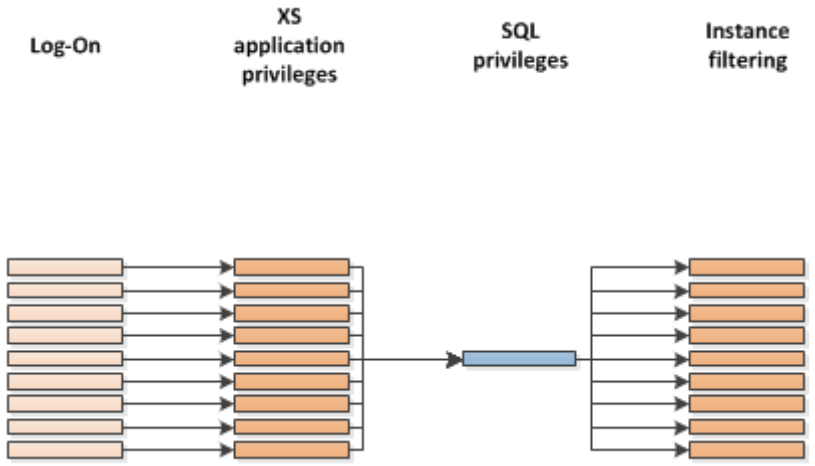
User handling in XS
Plain DB user

- **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**



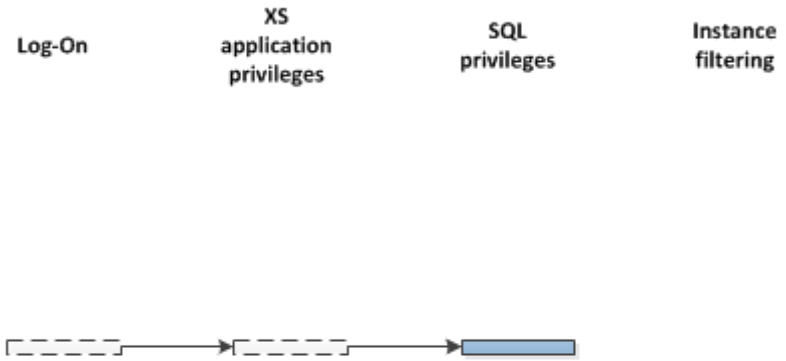
User handling in XS
SQLCC scenario (best practice for stand-alone XS Apps)

- **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 privilges
- **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**

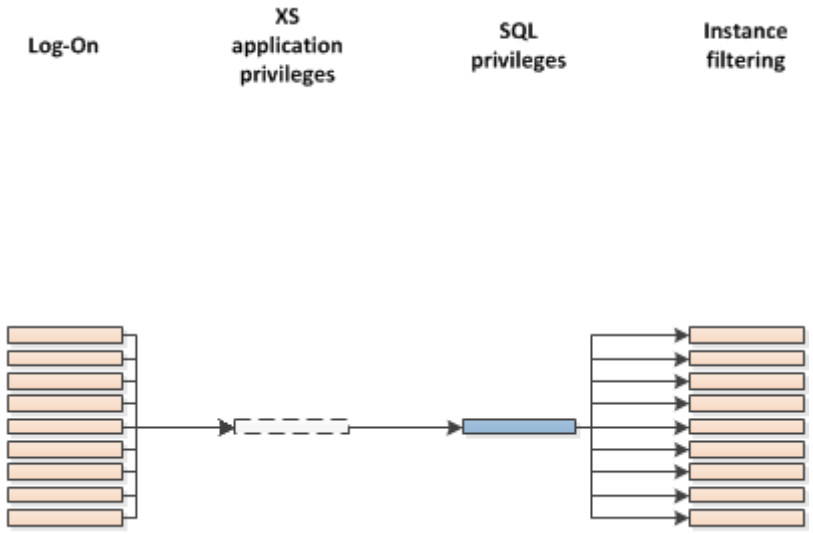


User handling in XS
Technical user scenario

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.getNString(1) + "\t" + rs.getNString(2) + "\t" +
                        rs.getNString(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.epmNe                    tem" ' +
            ' 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.getNString(1) + "\t" + rs.getNString(2) + "\t" +
                        rs.getNString(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.getNString(1) + "\t" + rs.getNString(2) + "\t" +
                        rs.getNString(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.getNString(1) + "\t" + rs.getNString(2) + "\t" +
                        rs.getNString(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                tem"' +
            ' 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 type specific

New database interface
Examples – new $.hdb interface

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

var query = 'SELECT * FROM "sap.hana.democo                    M.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";
   }
}
```

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

New database interface
Examples – new $.hdb interface

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

var query = 'SELECT * FROM "sap.hana.democontent.epm                    ' +
            ' 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

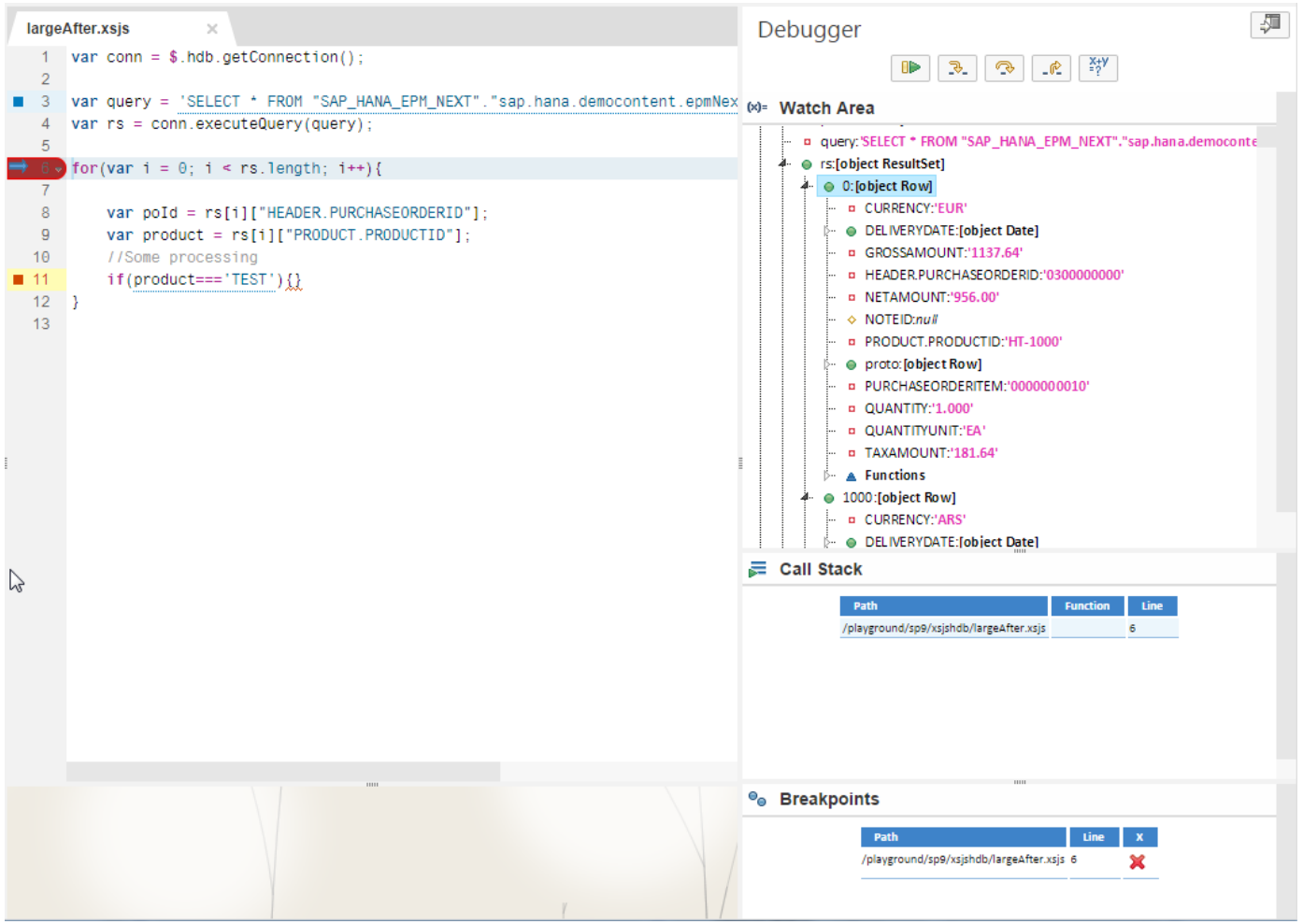
New database interface
Examples – new $.hdb interface

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

var query = 'SELECT * FROM "sap.hana.democontent.epmNext.data::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";
   }
}
```

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



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**

SMTP
(sending mails from xsjs application code)

ZIP and GZIP support (read and write)

XML parsing support

Secure Store for applications

Better Multipart Support

Asynchronous request completion

Text Access

Text Mining

Management of Scheduled Jobs

Various utility functions (hash, encrypt)

AntiVirus for applications

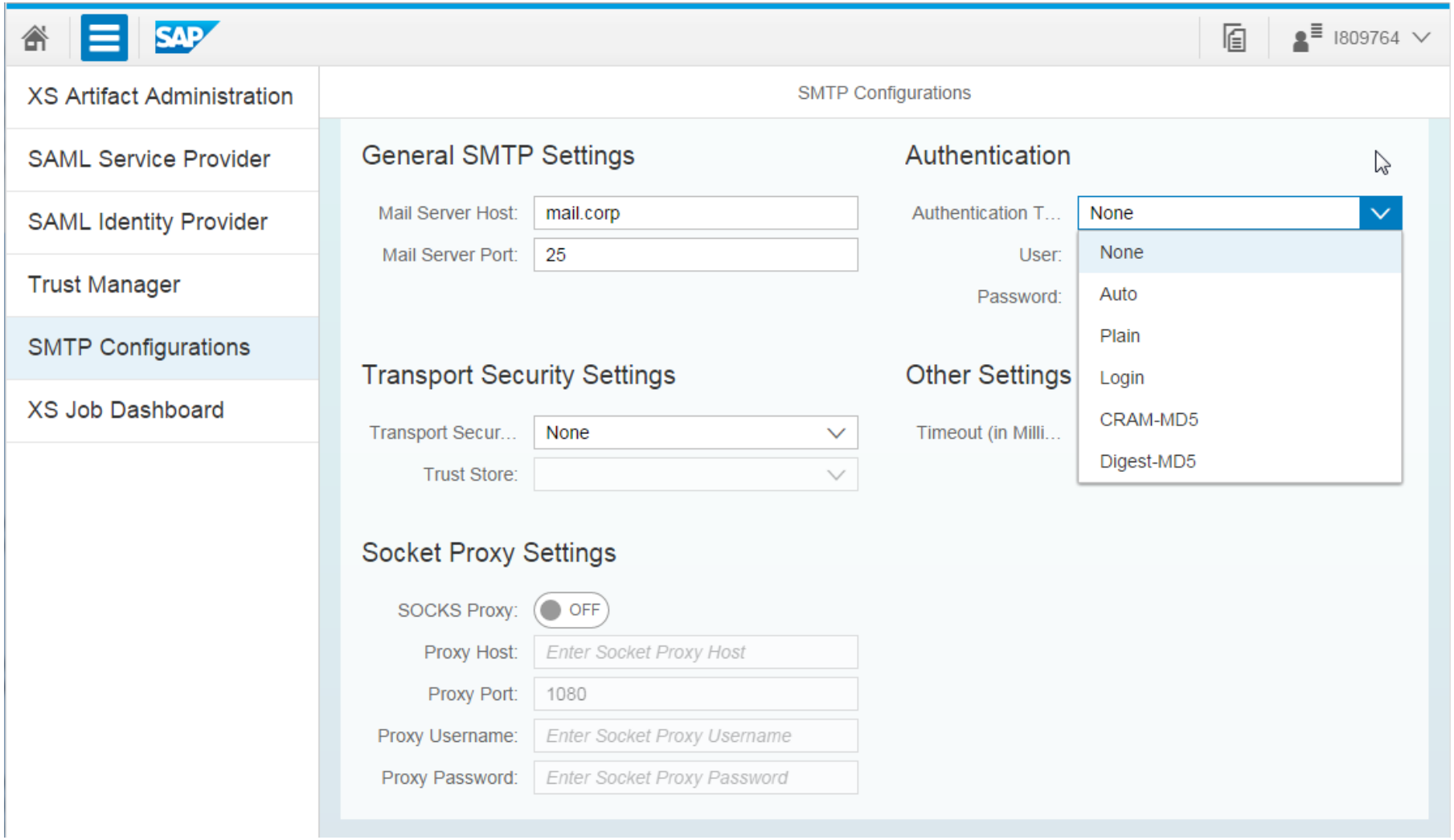
New XSJS-APIs
Examples – SMTP

```
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



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, atts) {
    startElementHandlerConcat += name;
    if (name === "note") {
        startElementHandlerConcat += " noteName = '" + atts.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

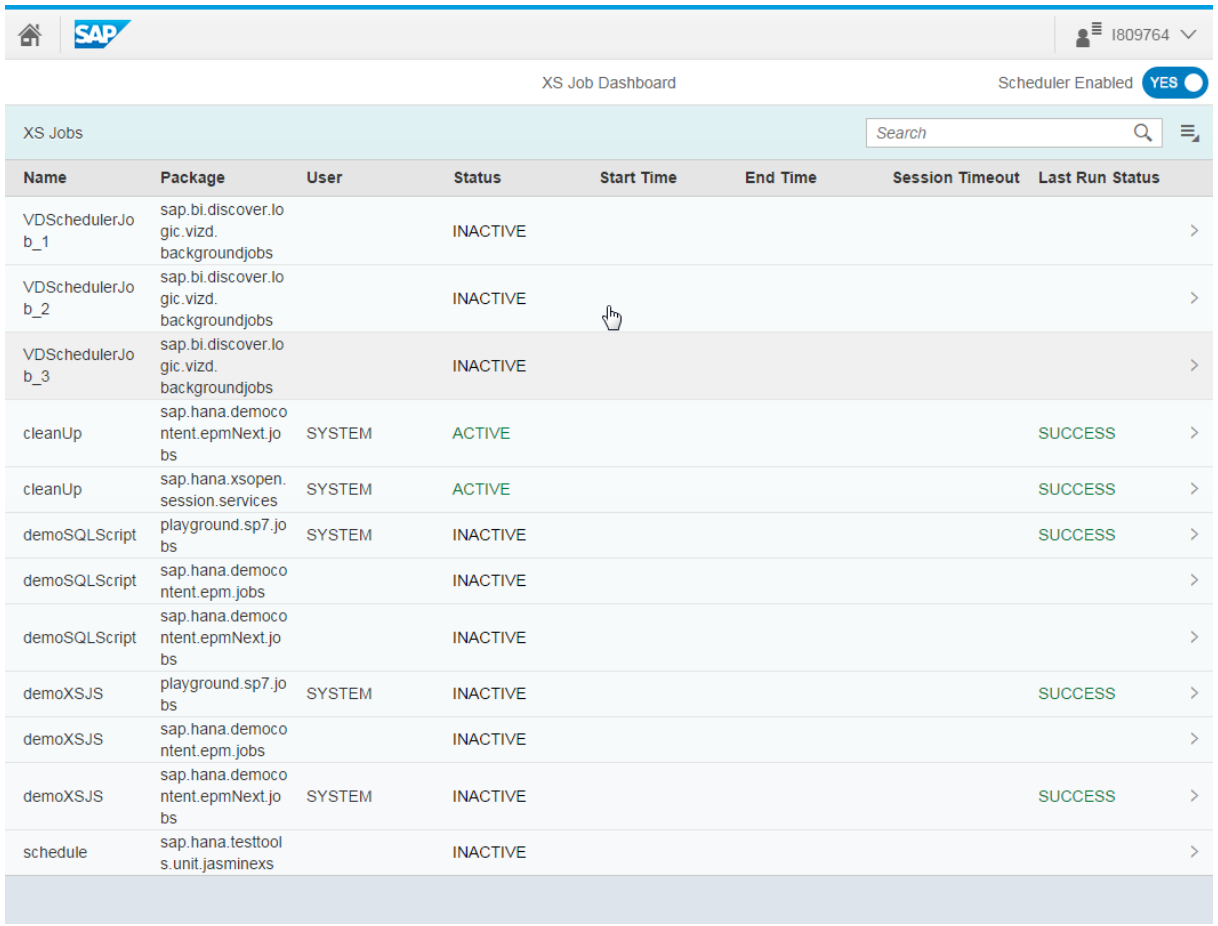
- Scalar parameters
- References to tables
- Creating local tables by type

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/

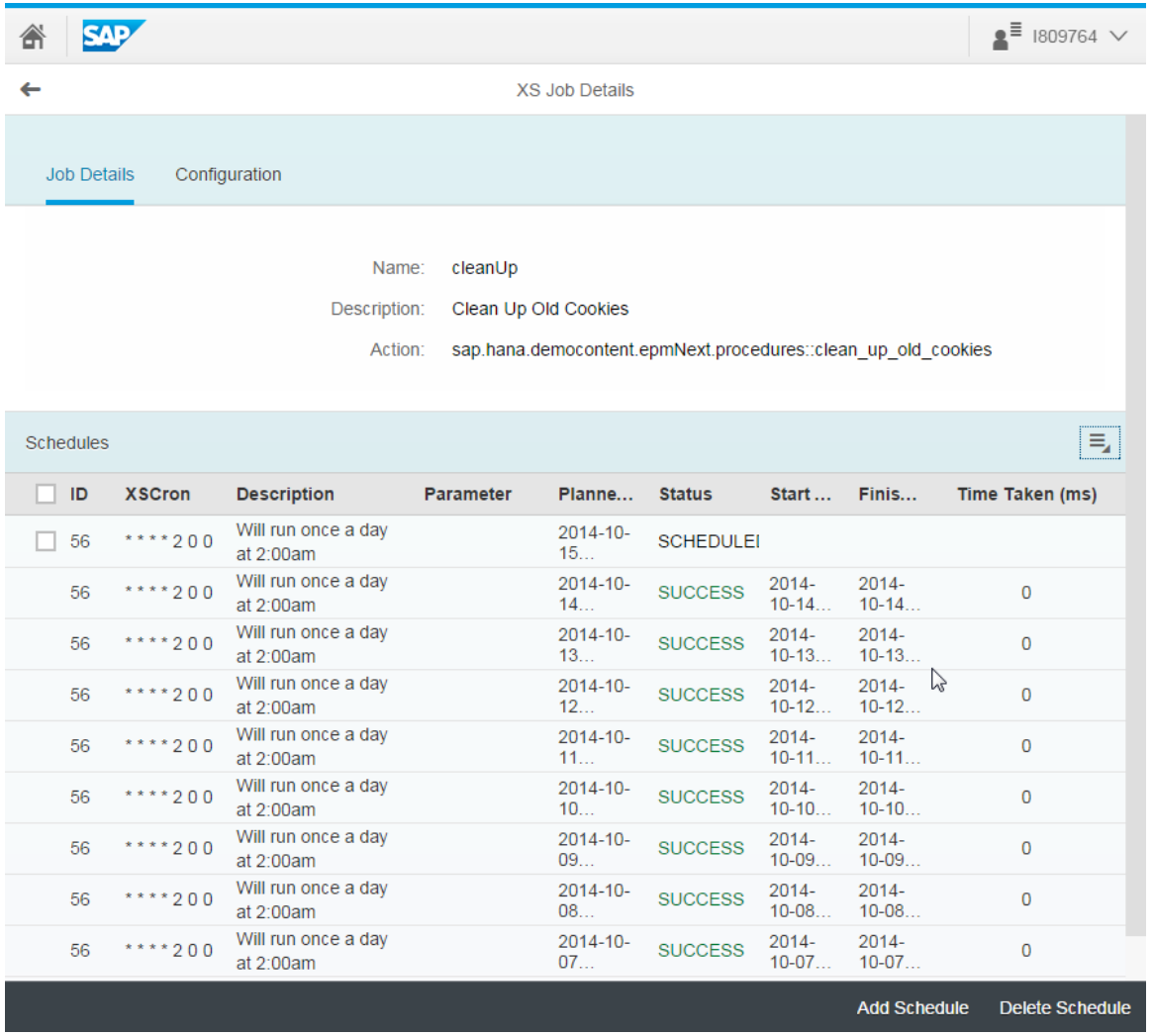


Name	Package	User	Status	Start Time	End Time	Session Timeout	Last Run Status
VDSchedulerJob_1	sap.bi.discover.logic.vizd.backgroundjobs		INACTIVE				
VDSchedulerJob_2	sap.bi.discover.logic.vizd.backgroundjobs		INACTIVE				
VDSchedulerJob_3	sap.bi.discover.logic.vizd.backgroundjobs		INACTIVE				
cleanUp	sap.hana.democontent.epmNext.jobs	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.epmNext.jobs		INACTIVE				
demoXSJS	playground.sp7.jobs	SYSTEM	INACTIVE				SUCCESS
demoXSJS	sap.hana.democontent.epm.jobs		INACTIVE				
demoXSJS	sap.hana.democontent.epmNext.jobs	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



OData

HANA XS SPS9: **OData**

Configurable Cache-settings for Metadata-Document

E-Tag support

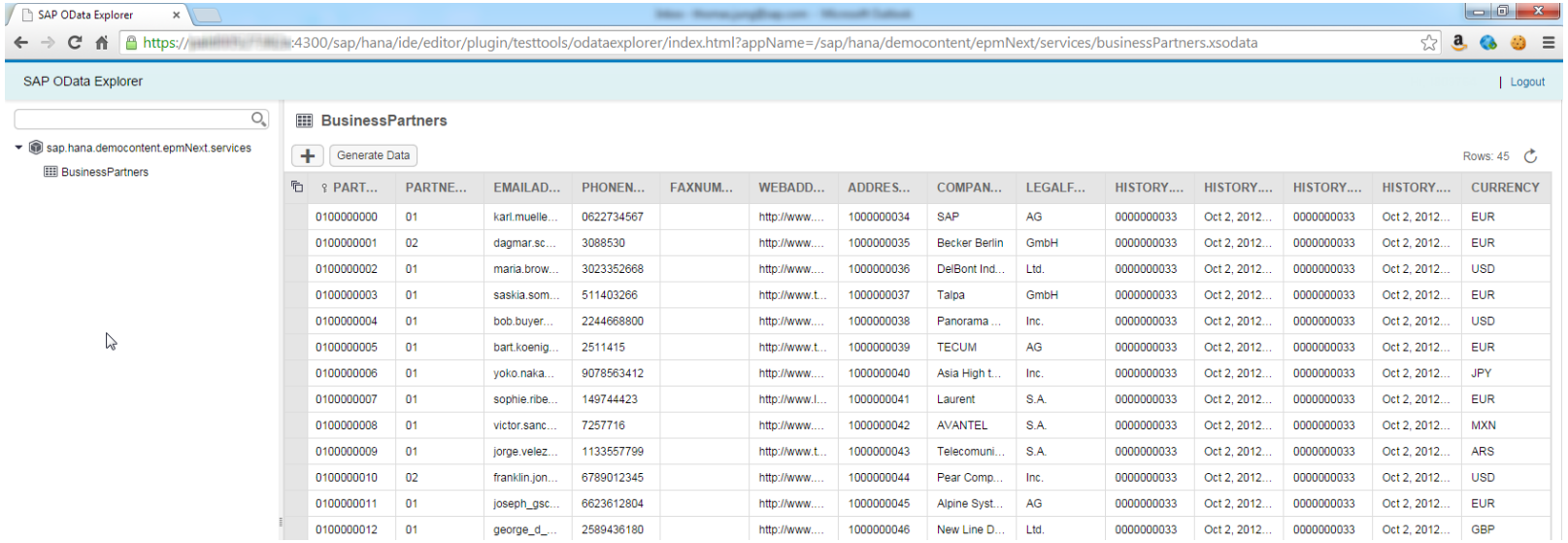
Automatic parallelization for GET batch requests

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>



Web Dispatcher

HANA XS SPS9: **Local Web Dispatcher**

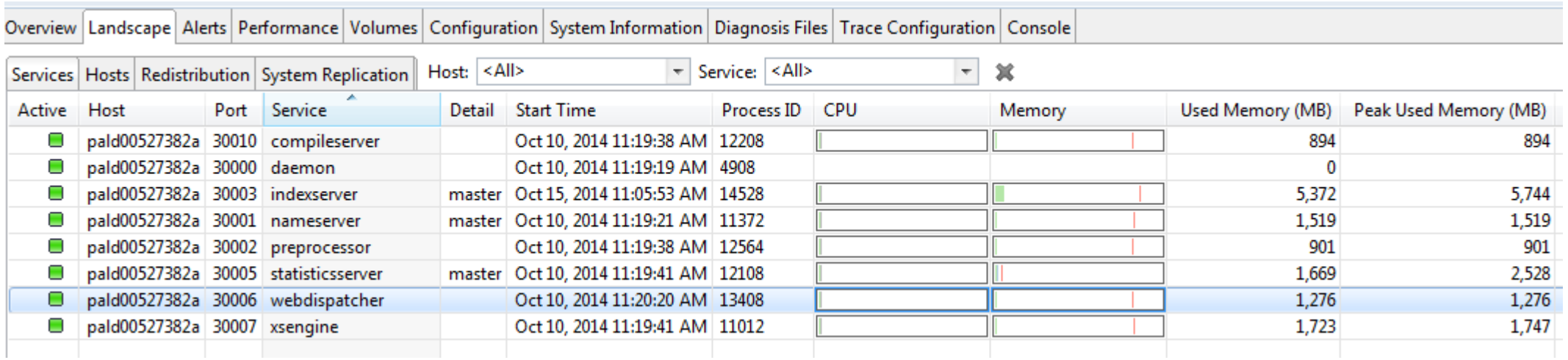
Full Integration into HANA Process Management

- Monitoring
- Configuration
- Administration

HANA XS SPS9: **Local Web Dispatcher**

Full Integration into HANA Process Management

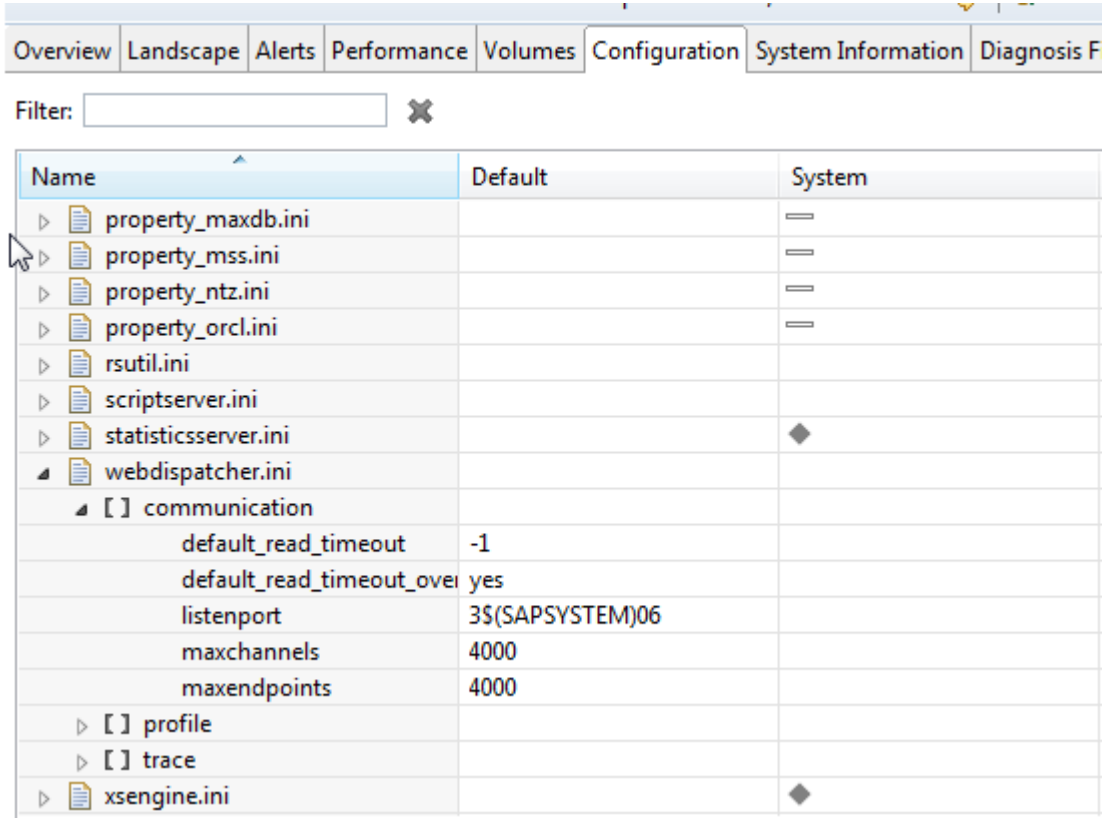
- Monitoring



HANA XS SPS9: **Local Web Dispatcher**

Full Integration into HANA Process Management

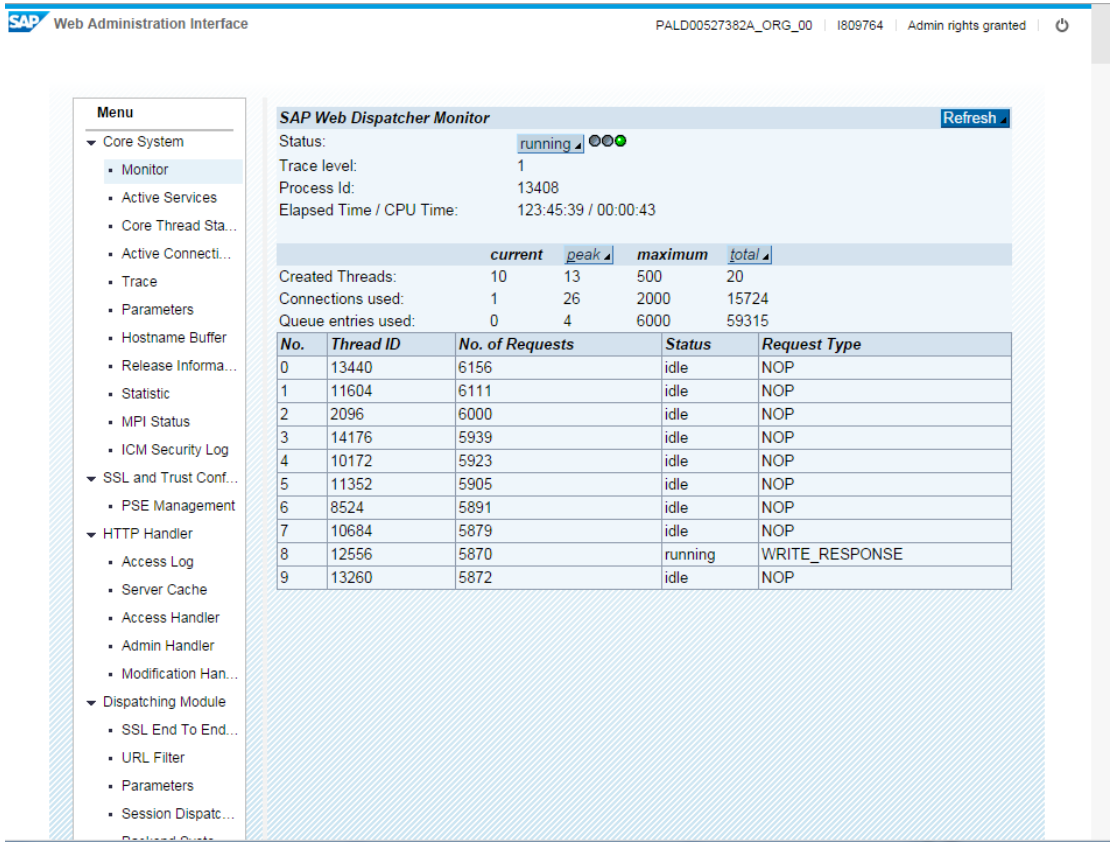
- Configuration



HANA XS SPS9: **Local Web Dispatcher**

Full Integration into HANA Process Management

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



XS Admin Tool

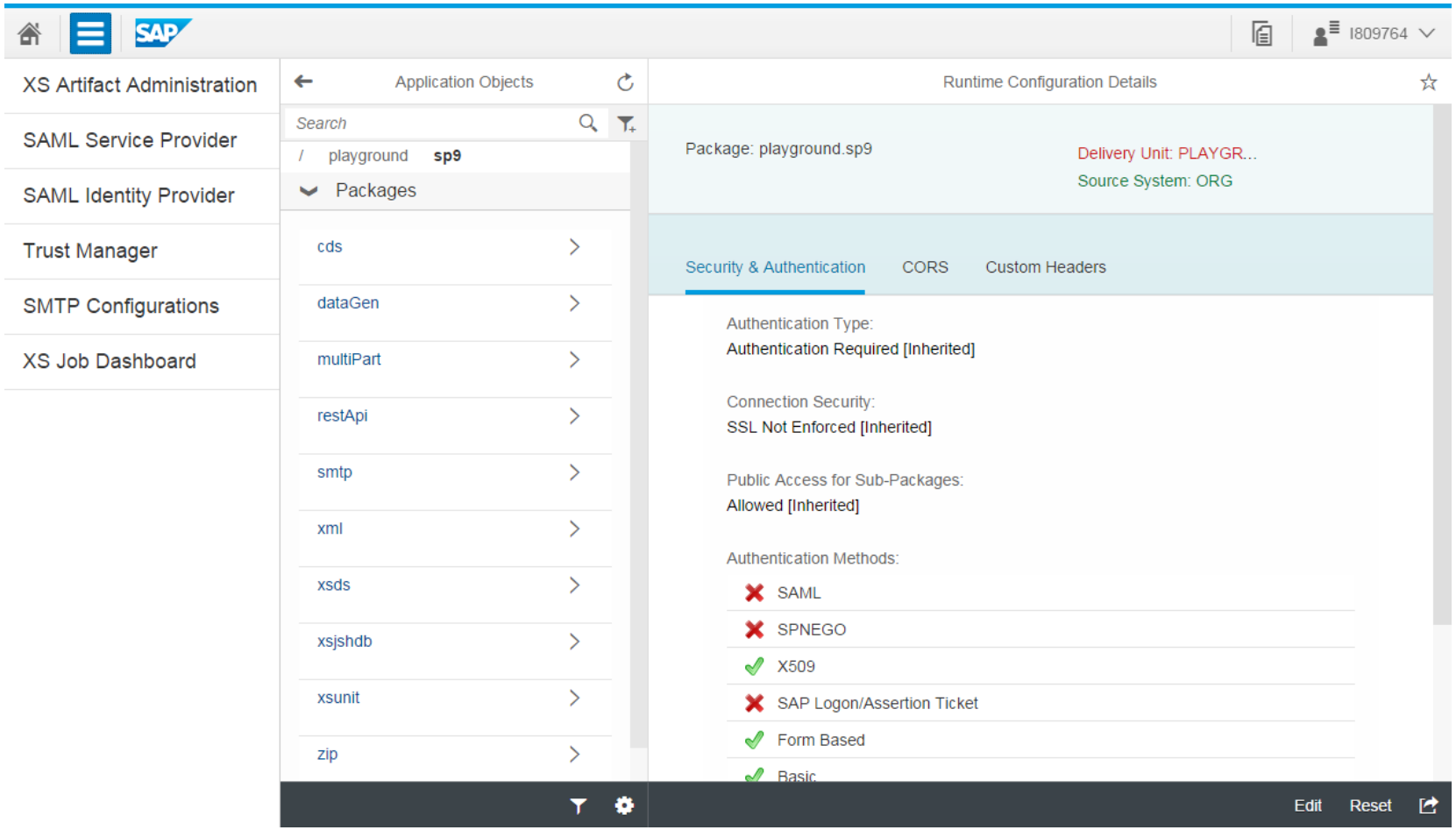
HANA XS SPS9: **HANA XS Admin Tool**

Much improved usability

User Self Service

UI of Logon Screen is Customizable

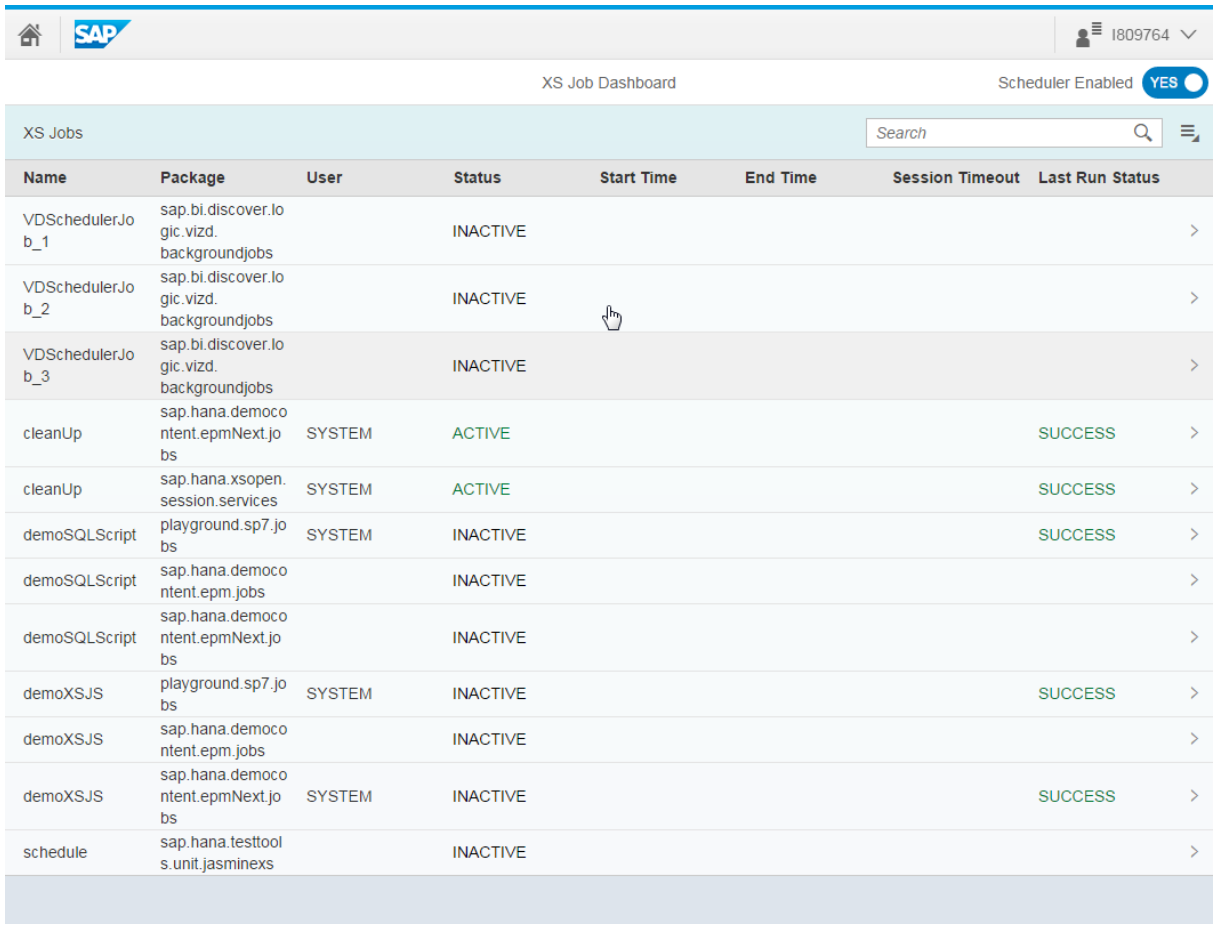
HANA XS SPS9: **HANA XS Admin Tool – new visual design**



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

New Management UI (XS Job Dashboard)

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

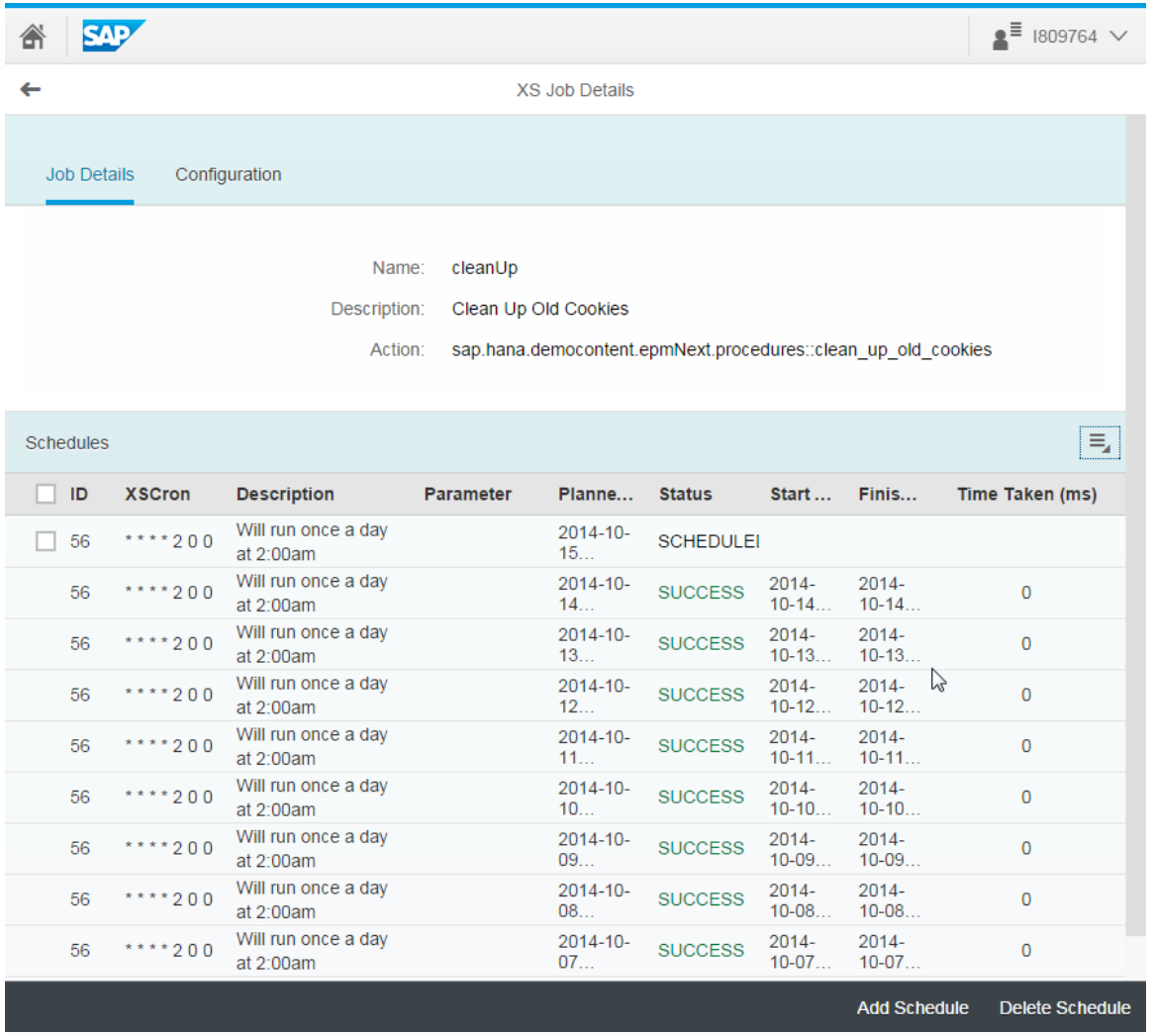


Name	Package	User	Status	Start Time	End Time	Session Timeout	Last Run Status
VDSchedulerJob_1	sap.bi.discover.logic.vizd.backgroundjobs		INACTIVE				
VDSchedulerJob_2	sap.bi.discover.logic.vizd.backgroundjobs		INACTIVE				
VDSchedulerJob_3	sap.bi.discover.logic.vizd.backgroundjobs		INACTIVE				
cleanUp	sap.hana.democontent.epmNext.jobs	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.epmNext.jobs		INACTIVE				
demoXSJS	playground.sp7.jobs	SYSTEM	INACTIVE				SUCCESS
demoXSJS	sap.hana.democontent.epm.jobs		INACTIVE				
demoXSJS	sap.hana.democontent.epmNext.jobs	SYSTEM	INACTIVE				SUCCESS
schedule	sap.hana.testtools.unit.jasminexs		INACTIVE				

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

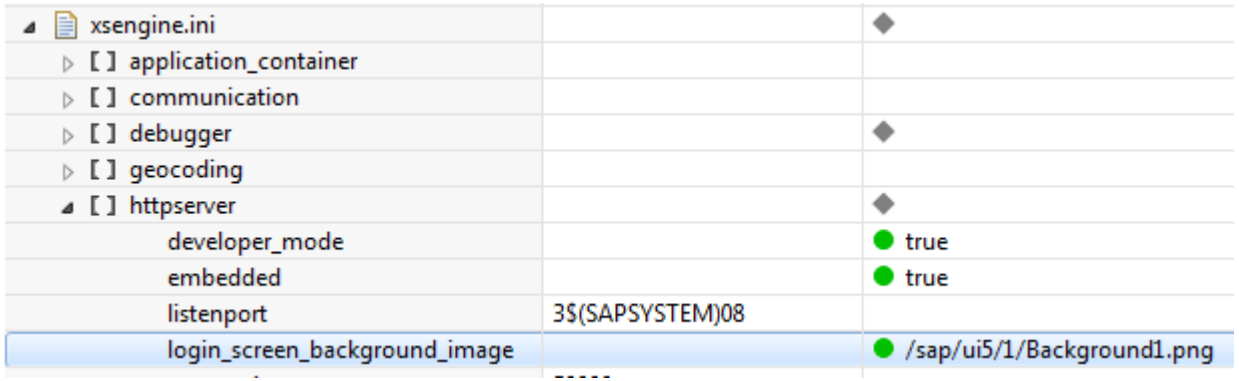
New Management UI (XS Job Dashboard)

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



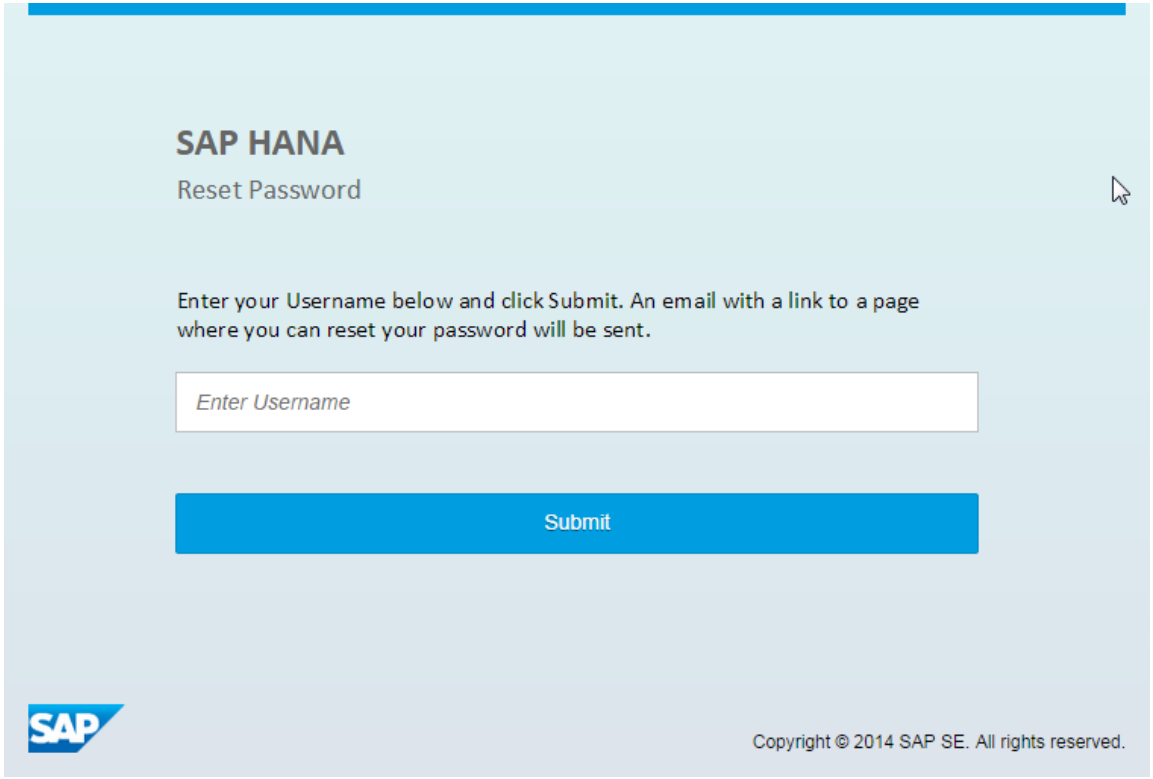
HANA XS SPS9: **Customize Login Screen**

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



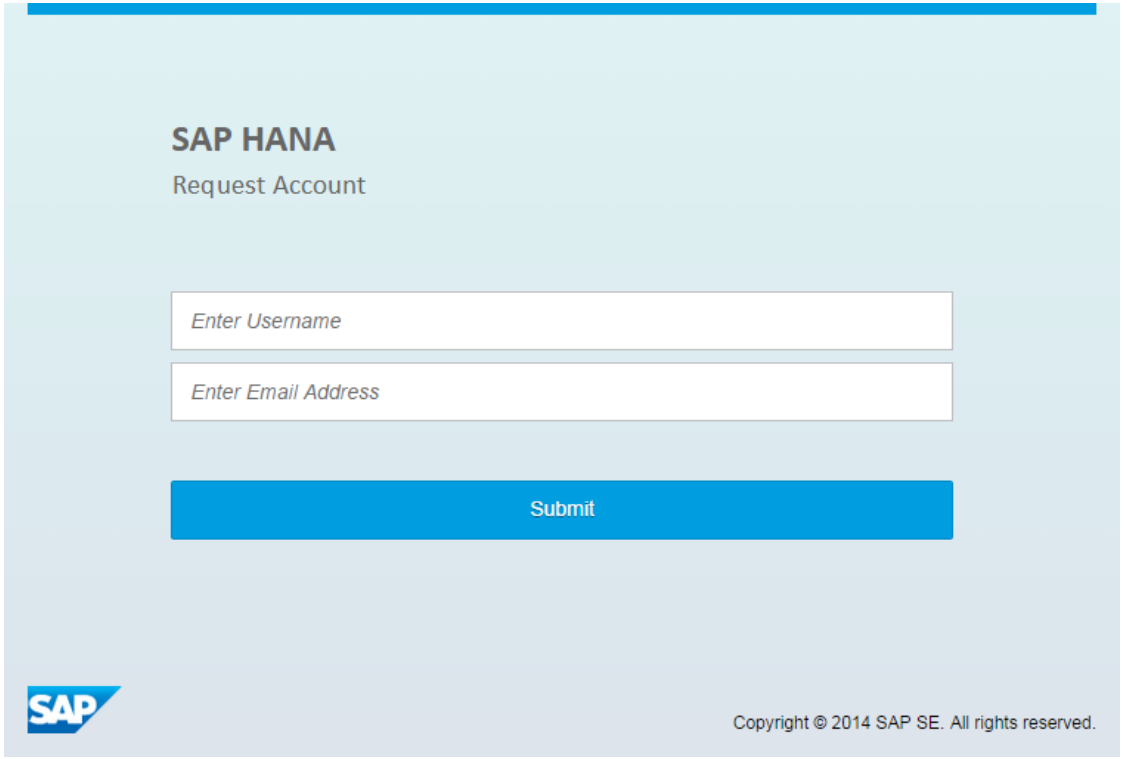
HANA XS SPS9: **User Self Service**

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



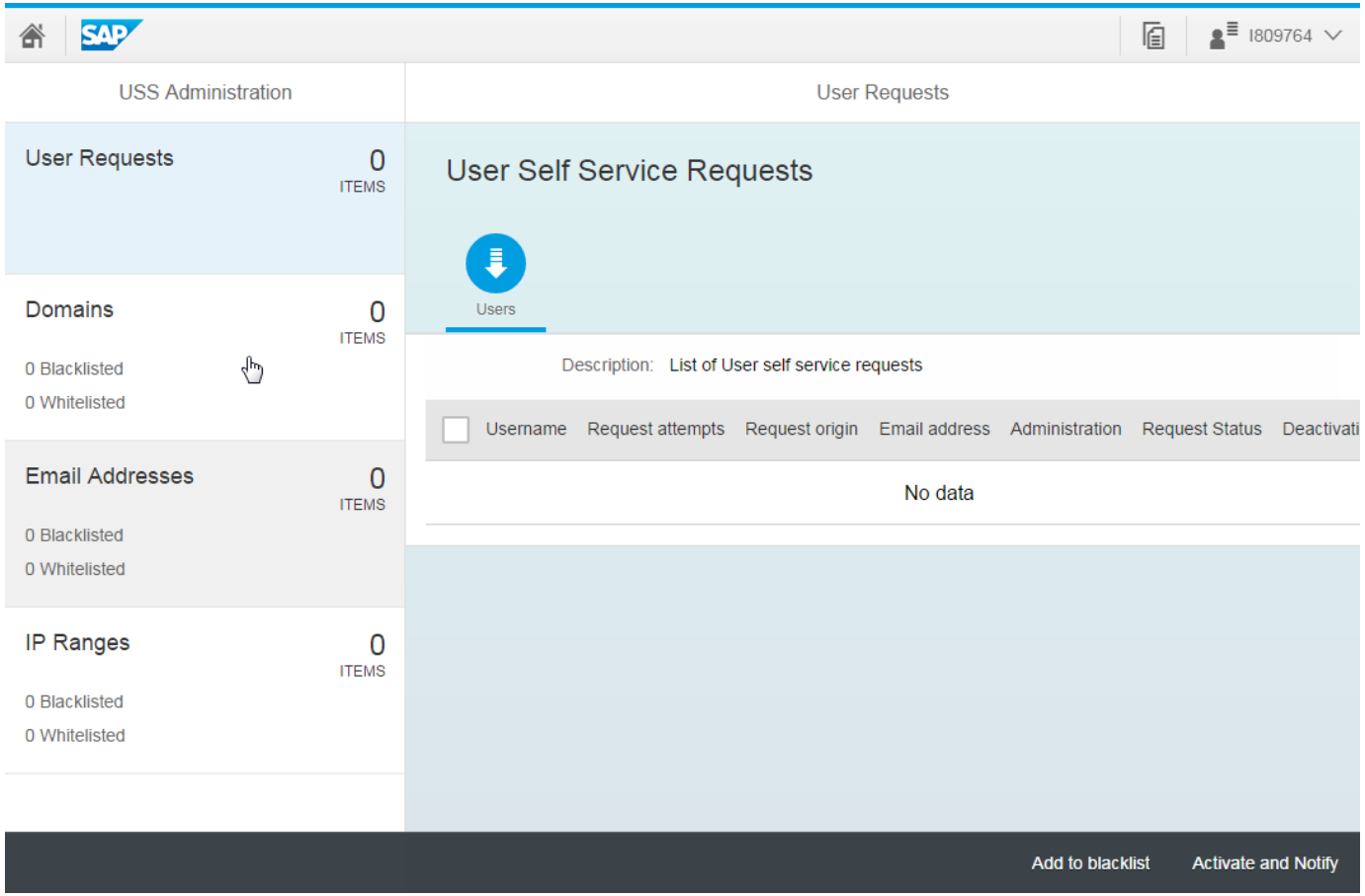
HANA XS SPS9: **User Self Service**

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



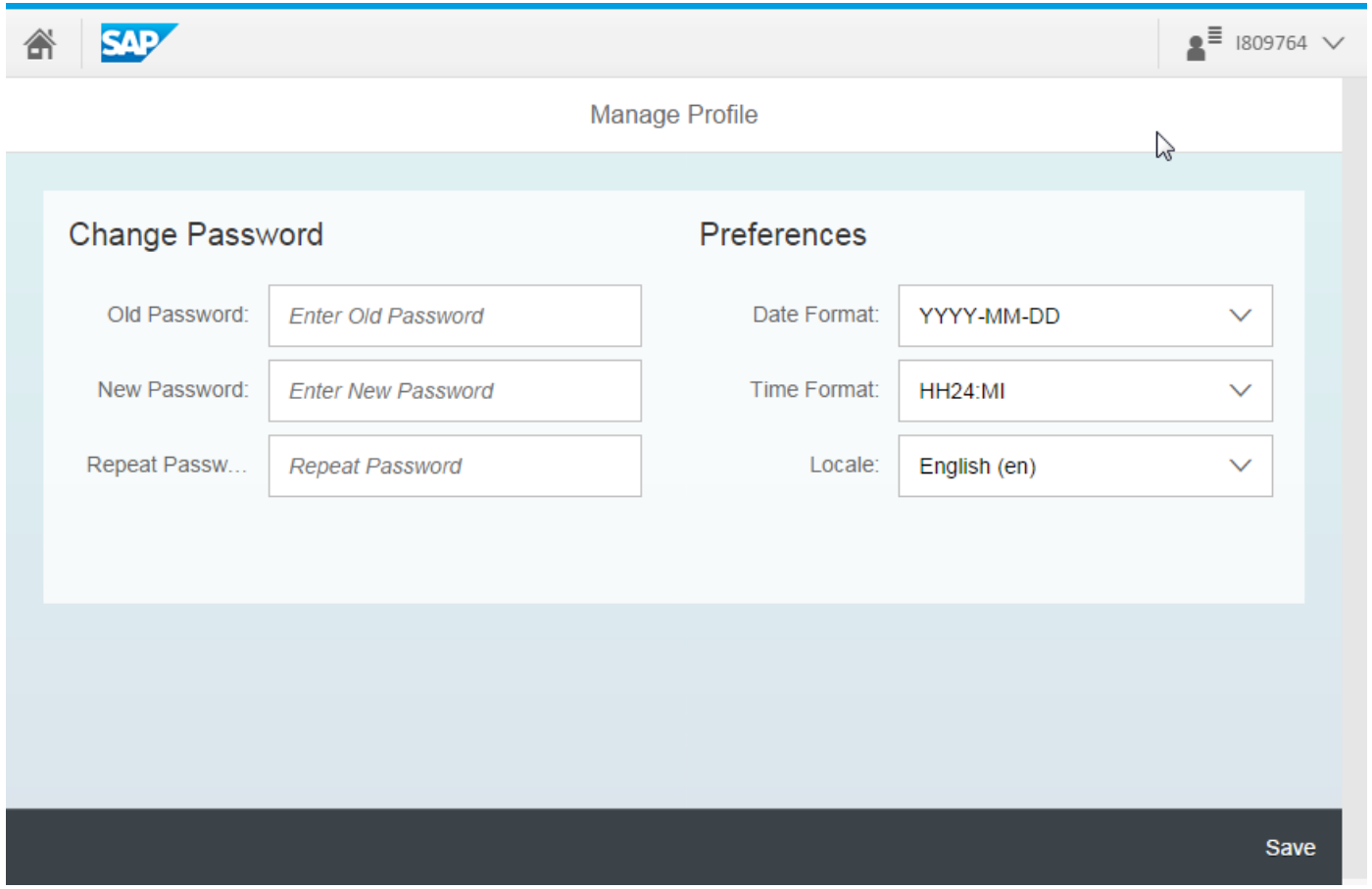
HANA XS SPS9: **User Self Service – Admin**

/sap/hana/xs/selfService/admin/



HANA XS SPS9: **User Self Service – User Profile**

/sap/hana/xs/formLogin/profile/



HANA Test Tools

HANA XS SPS9: **Test Framework**

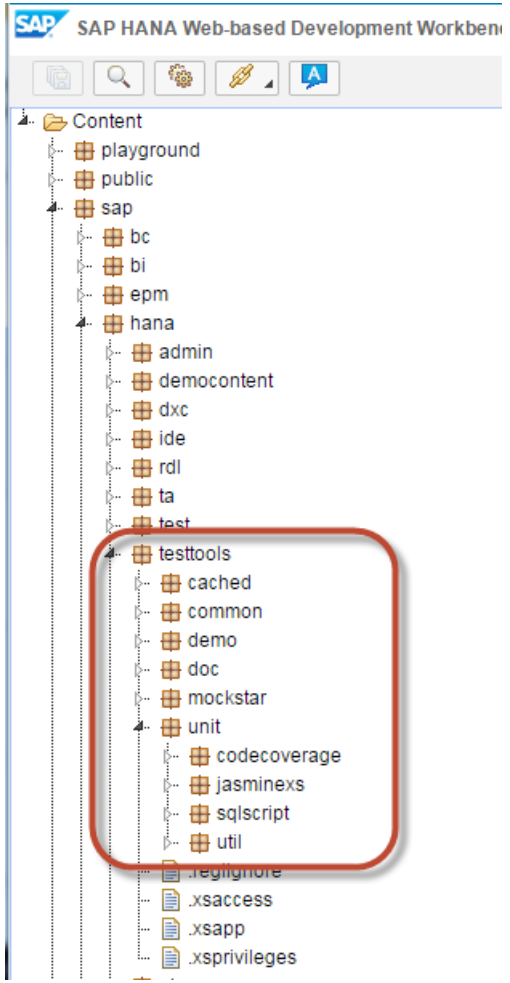
Unit Test Framework

Mock Framework

(shipped with HANA, but not pre-installed)

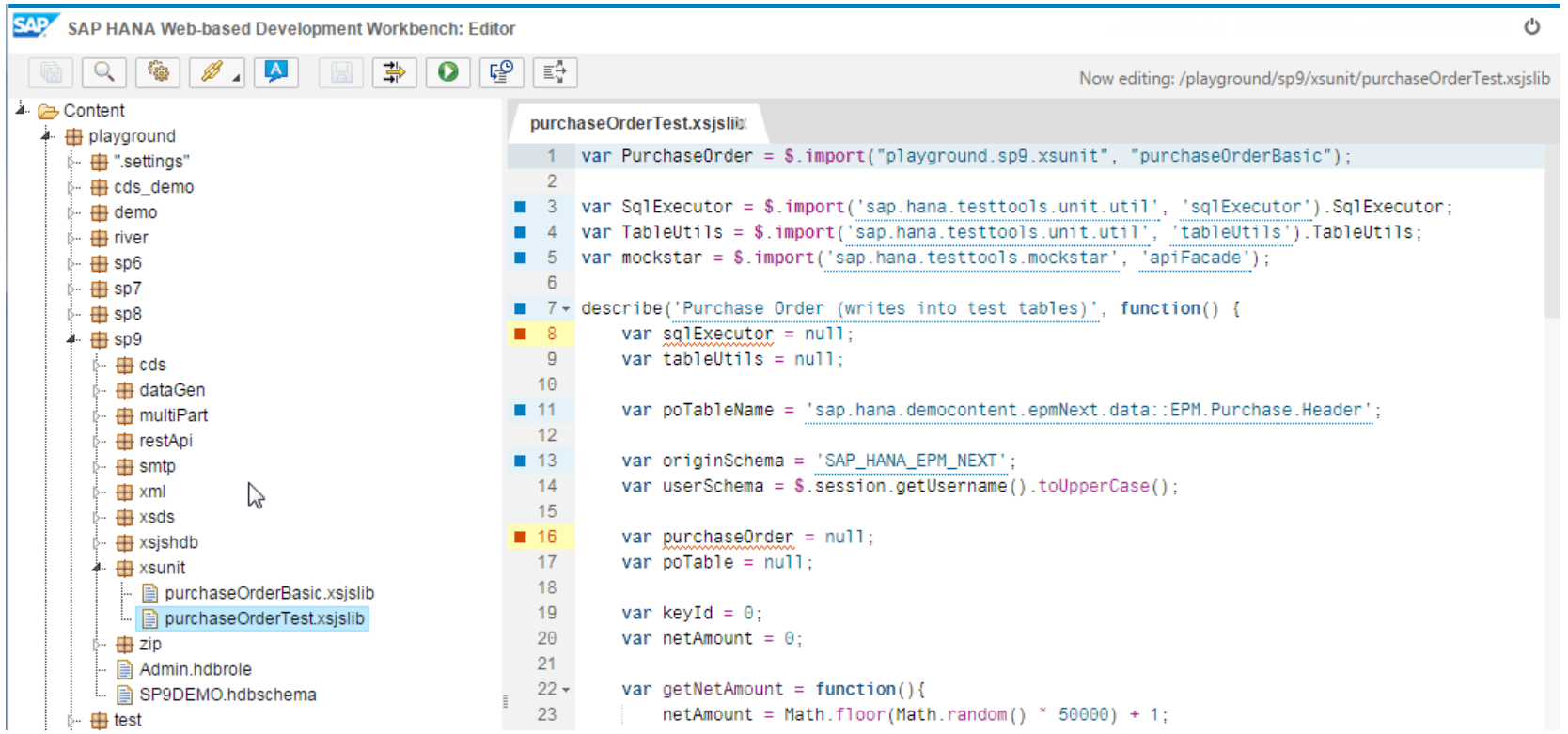
Contents of HANA_TEST_TOOLS Delivery Unit

- 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



Execute XSUnit Test in the browser

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

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

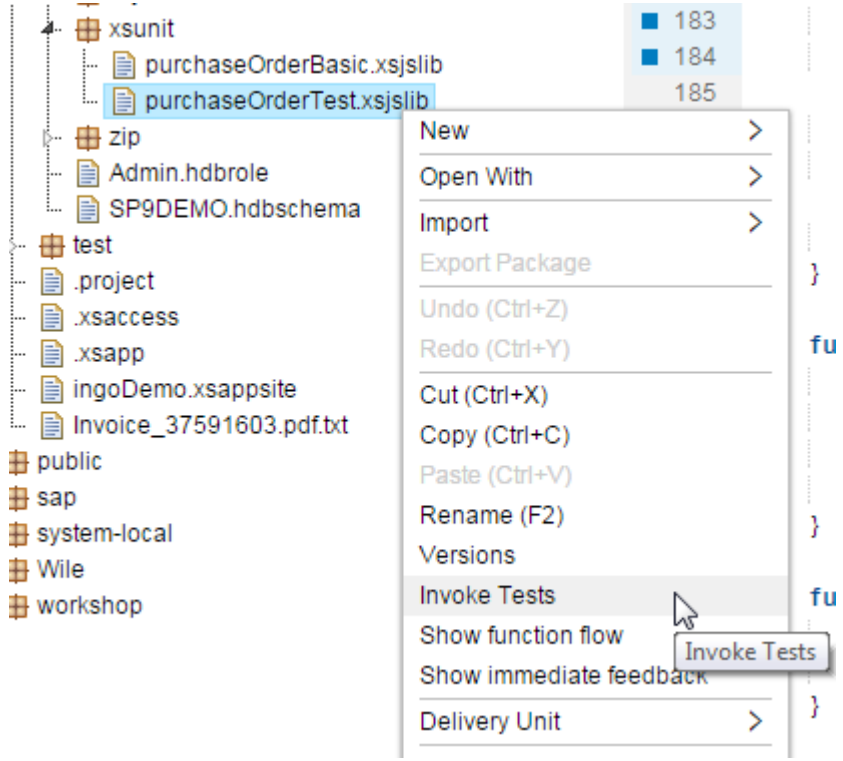
Execute XSUnit Test in the browser: URL Parameters (Continued)

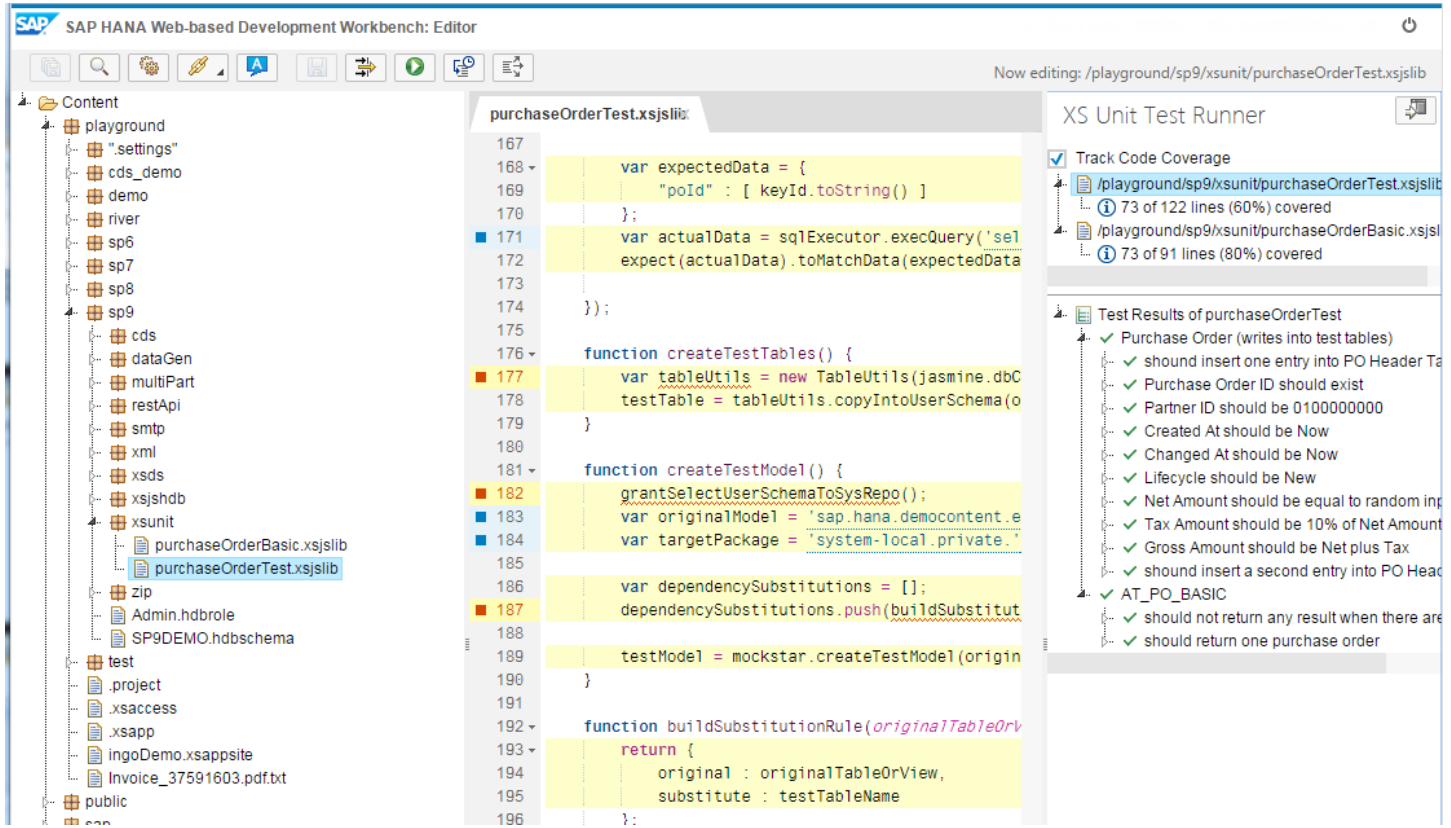
Parameter	Required	Description	Example
pattern	no	Naming pattern that identifies the .xsjslib files that contain the tests. If not specified, the pattern “*Test” is applied. You can use wildcards “?” and “*” to match a single and multiple arbitrary characters respectively.	pattern=Test* would match all 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.testtools.demo

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

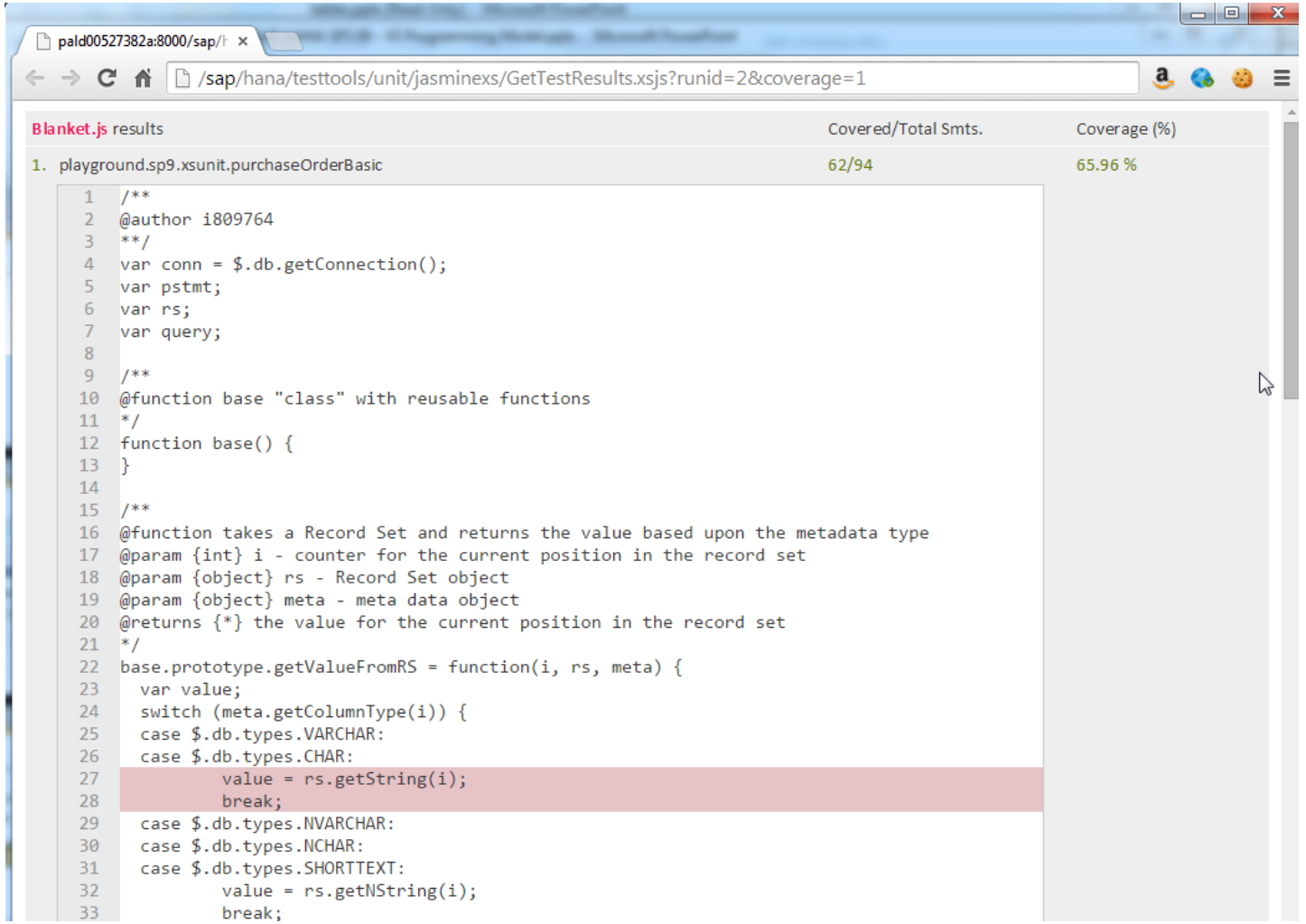
Parameter	Required	Description	Example
format	no	Specifies the output format the test runner uses to report test results. By default, the results will be reported as HTML document. This parameter has no effect if a custom reporter is provided via parameter “reporter”.	format=html shows results in HTML; 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

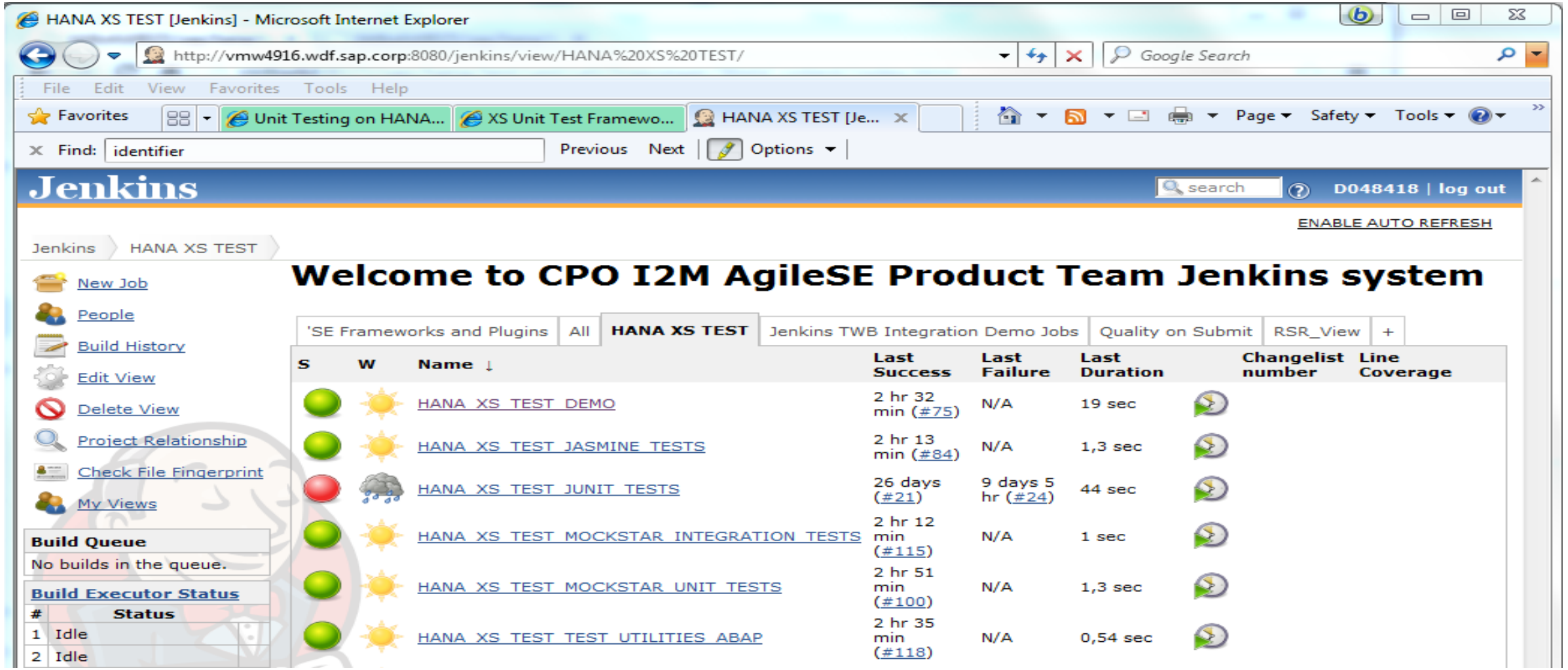




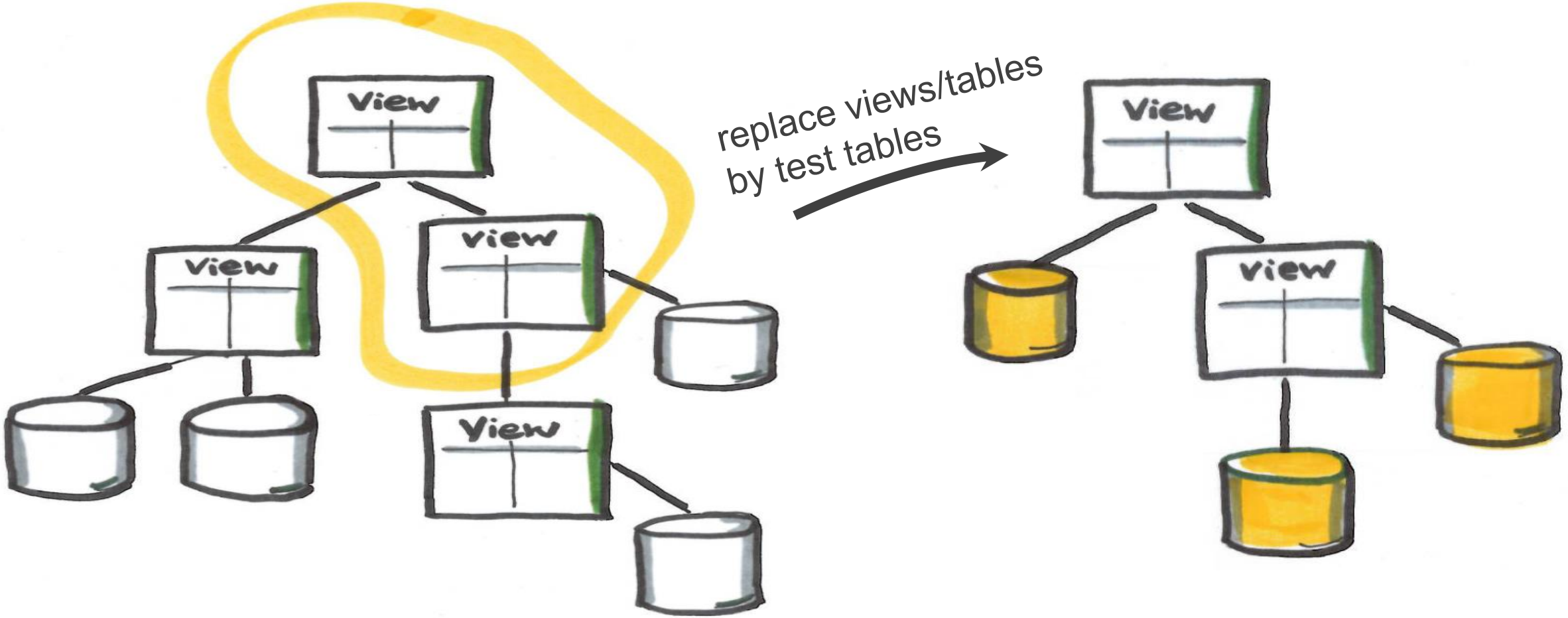
Analyze Test Coverage



View XSUnit Test Results



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

CDS
Lifecycle Management Example

SQL:

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

Adding new field:

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

CDS
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**

CDS
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);
    };
  };
};
```

CDS
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_GEOMETRY; };
```

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);
  officeId : Integer; };

entity Room {
  key id : Integer;
  inhabitants : Association[*] to Employee on inhabitants.officeId = 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; };
```

CDS
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);
};
```

CDS
Many to Many relationships

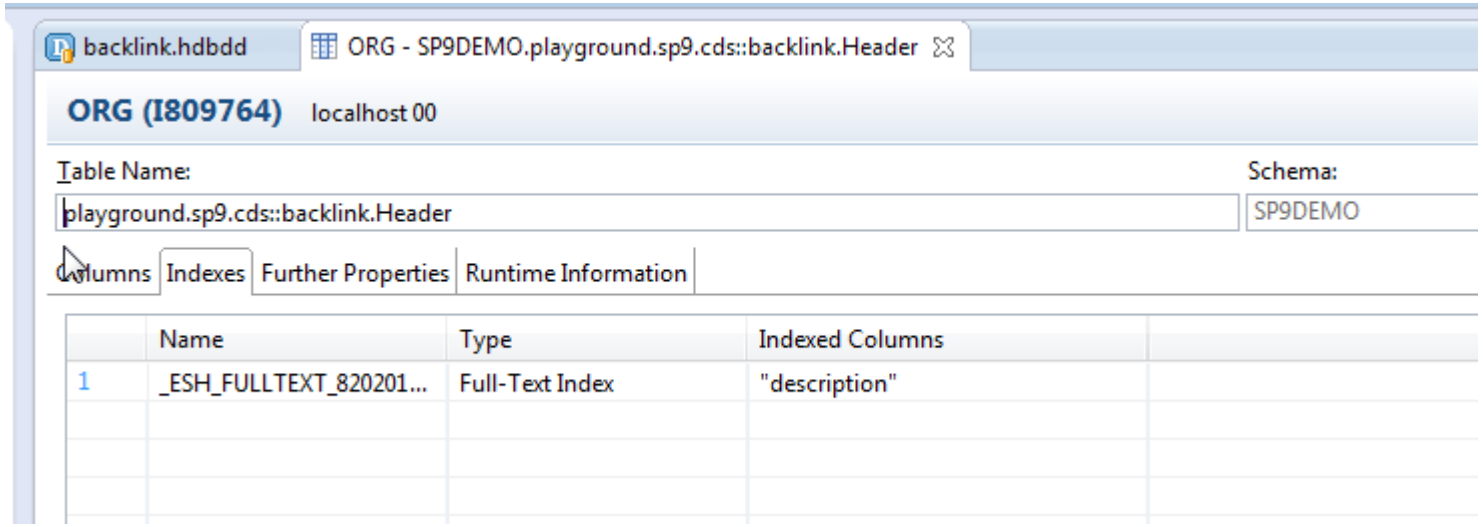
```
entity Employee {
  key id : Integer;
  name : String(80);
  projectLinks : Association[*] to E2P on projectLinks.e_id = id;  };

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;  };
```

CDS
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 };
```

CDS
Full Text Index Annotation

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



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

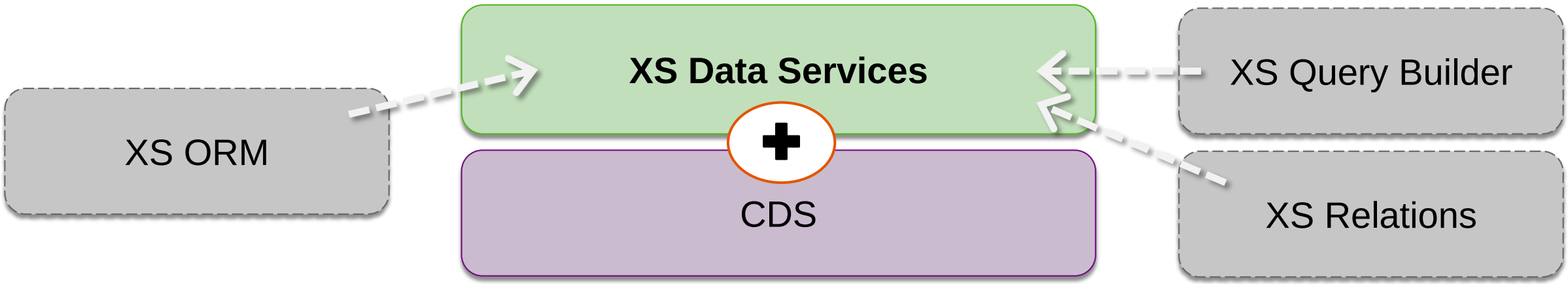
XSDS: Native CDS consumption

- ✓ 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

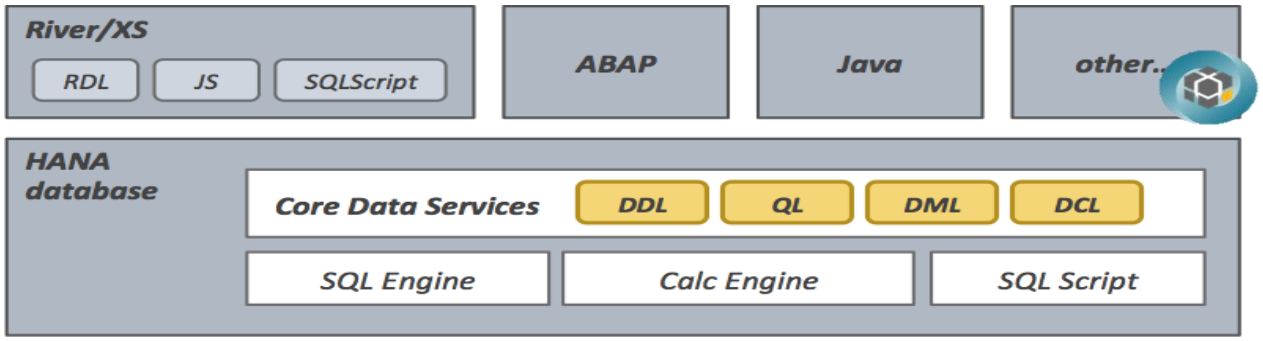


Extended Programming Model
XSDS – Data Models

Core Data Services

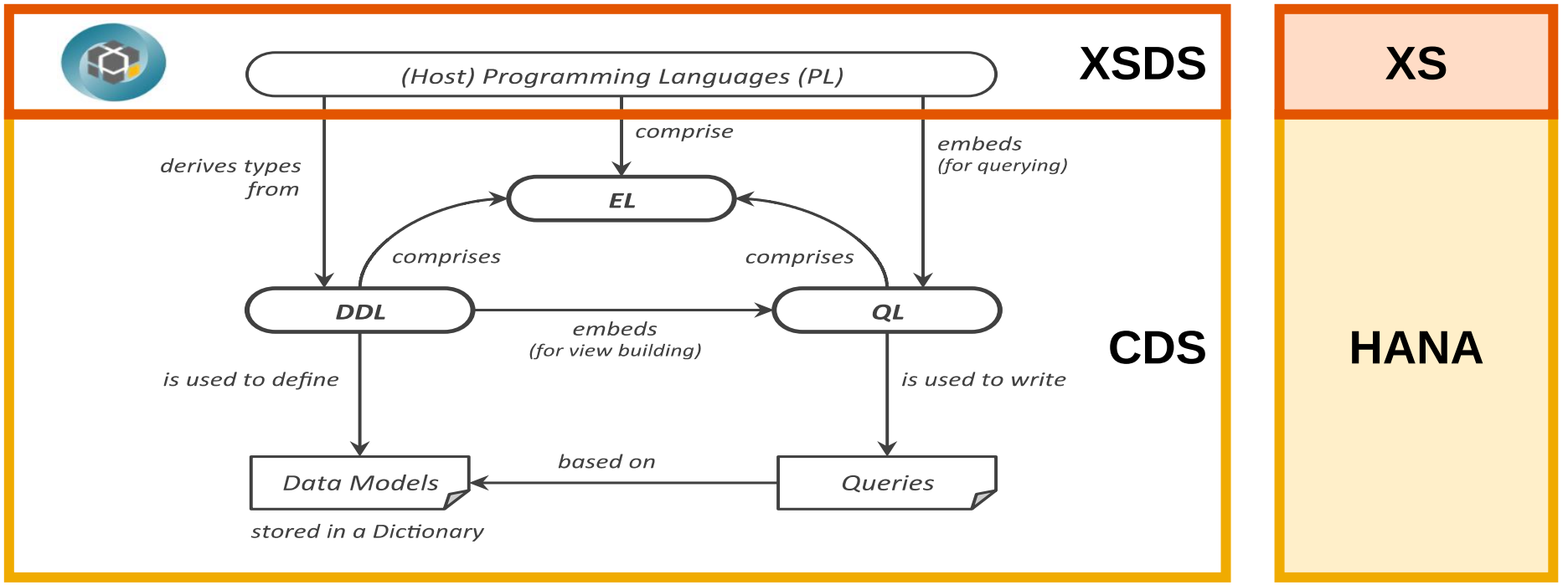
Core Data Services (CDS) are a cross-platform set of concepts and tools to define ***semantically rich data models*** for SAP HANA applications. A central part of CDS is the ***object-relational mapping*** that links semantically meaningful ***entities*** to their technical representation as ***records***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

Pre-generation of imports moves metadata processing overhead to design time

Import pre-generation

CDS import costly operation

XS Engine has no session state

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

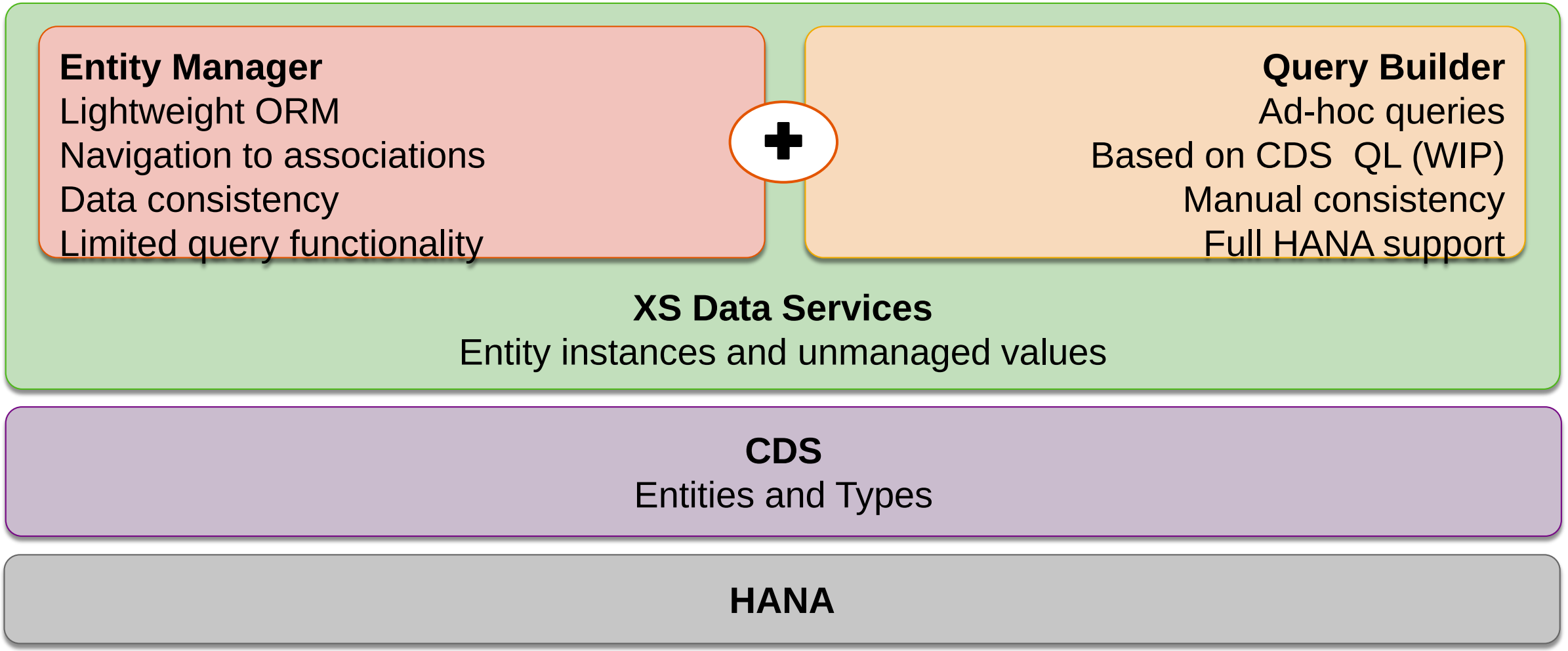
```
var User = XSDS.$importEntity("demo.bboard", "bboard.user");
```

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



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

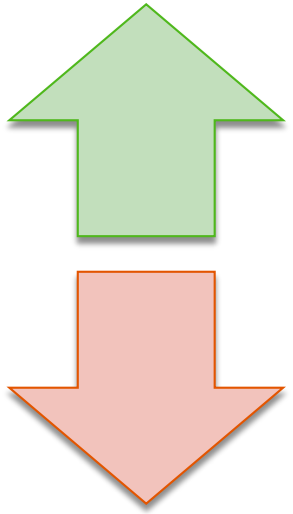
Single consistent view on every instance (enforced by $persist)

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

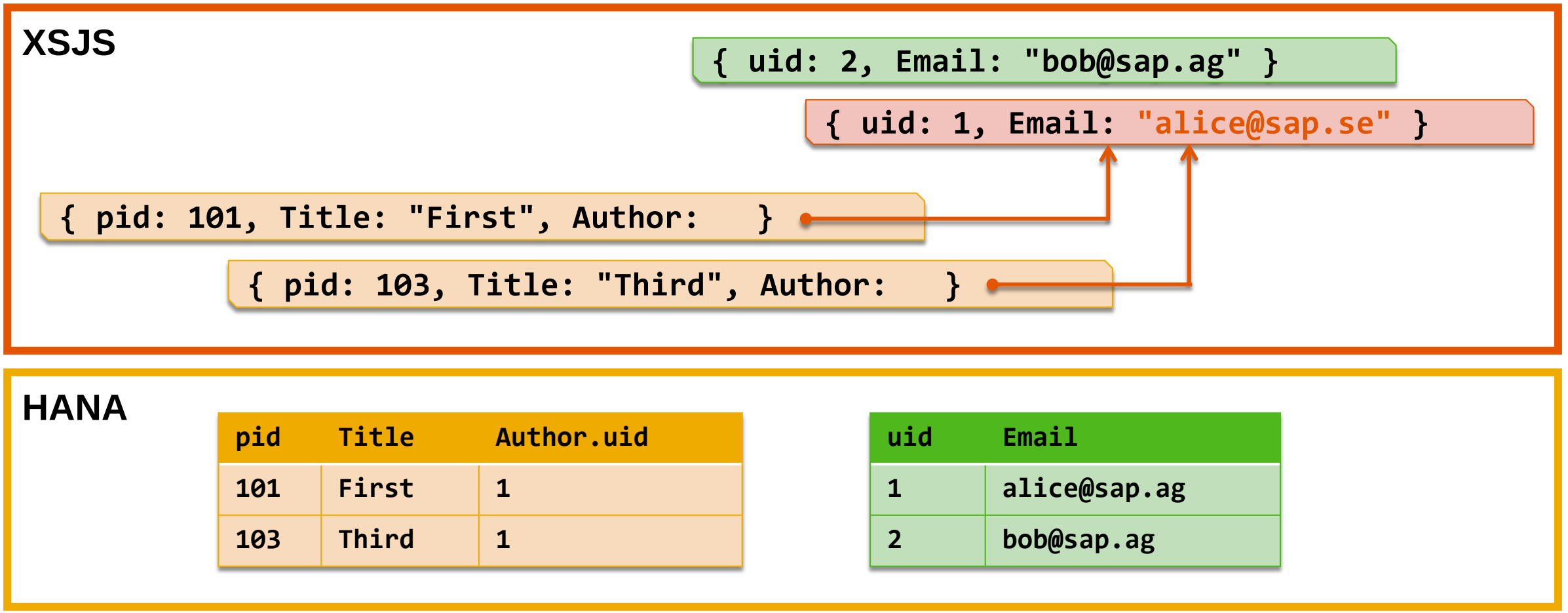
Downside: Entity cache

Additional overhead for instance management

Only limited subset of database functionality for instance retrieval



Consistent instance view



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();
```

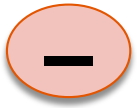
translated to:

```
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

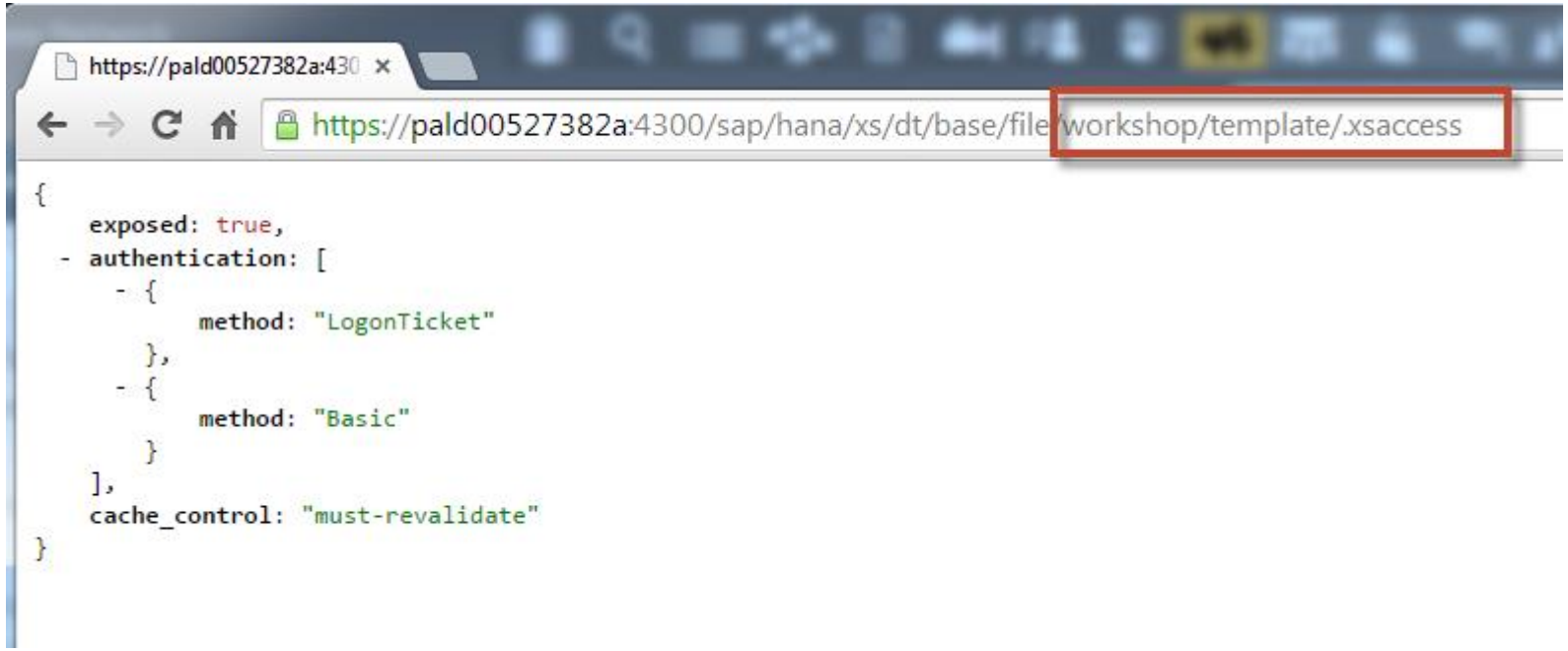
A standard reusable library shipped by SAP to simplify Stored Procedure calls from XSJS and make they “feel” like JavaScript functions

```
//*******************************************
//     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.**



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

- Enables you to browse and manipulate files and directories via HTTP
- Basic HTTP methods GET, PUT, and POST to send requests
- JSON is used as the default representation format
- Can also retrieve metadata and previous versions of the content

SAP HANA REST API – Workspace API

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

SAP HANA REST API – Transfer API

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

SAP HANA REST API – Metadata API

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

SAP HANA REST API – Change Tracking API

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

SAP HANA REST API – OData Services

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

- 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

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
    },
```

SAP River

SAP River – What's New in SPS 09?

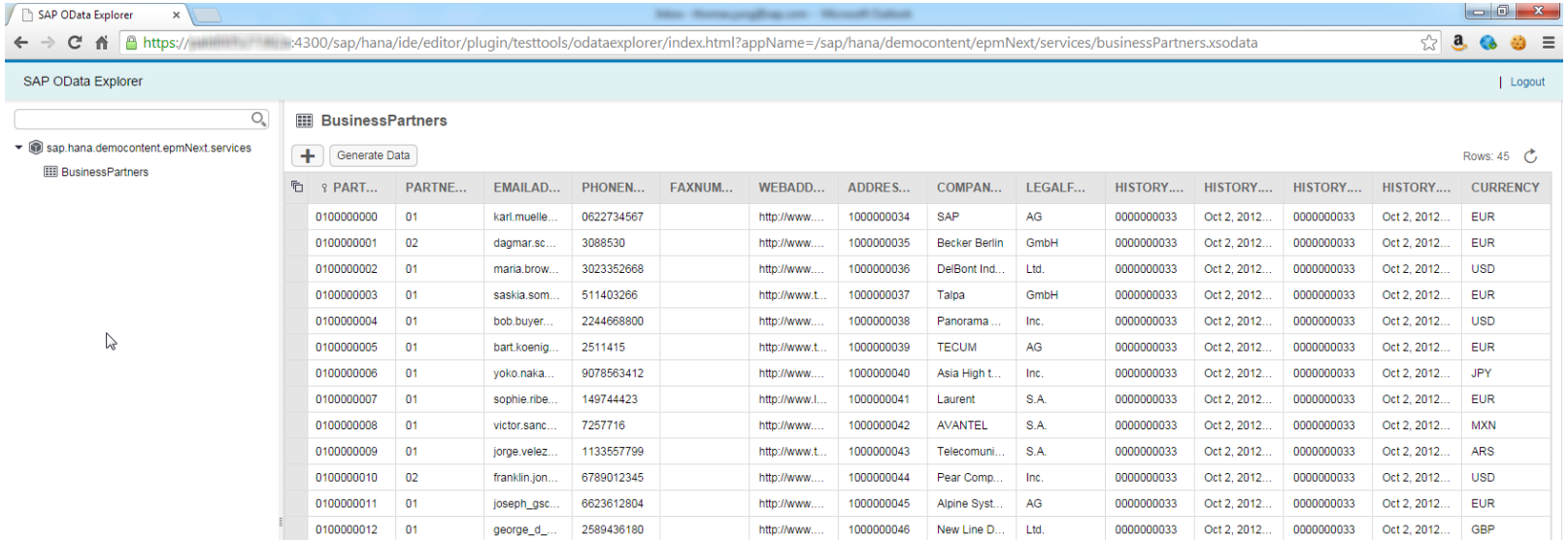
The benefits of the SAP River language (RDL) application development environment were validated by partners and early adaptors. However, based on feedback we received, and consistent with our strategic direction of building open and standard environments, SAP has decided to abandon a proprietary language 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

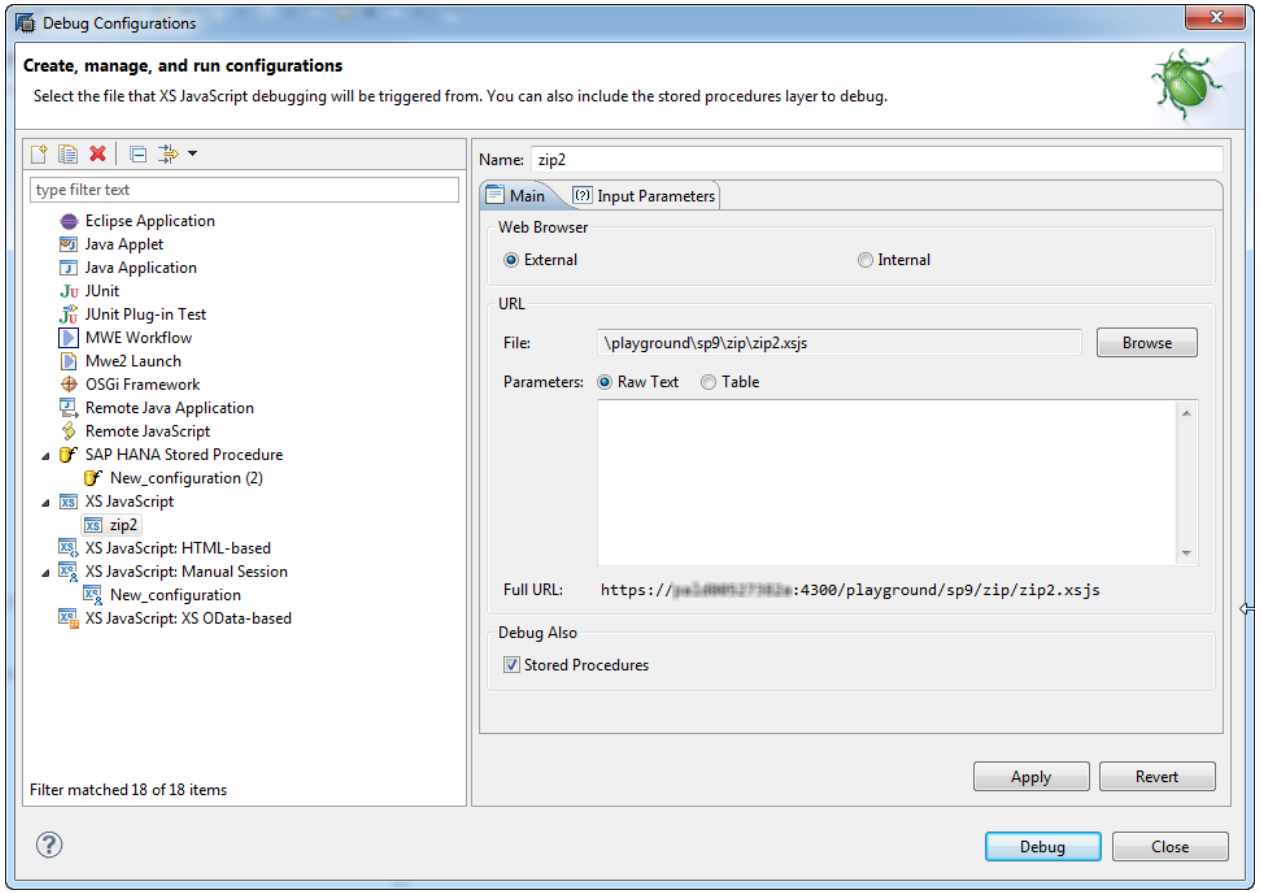
General OData test and data generation tool which supports XSODATA services



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");
```

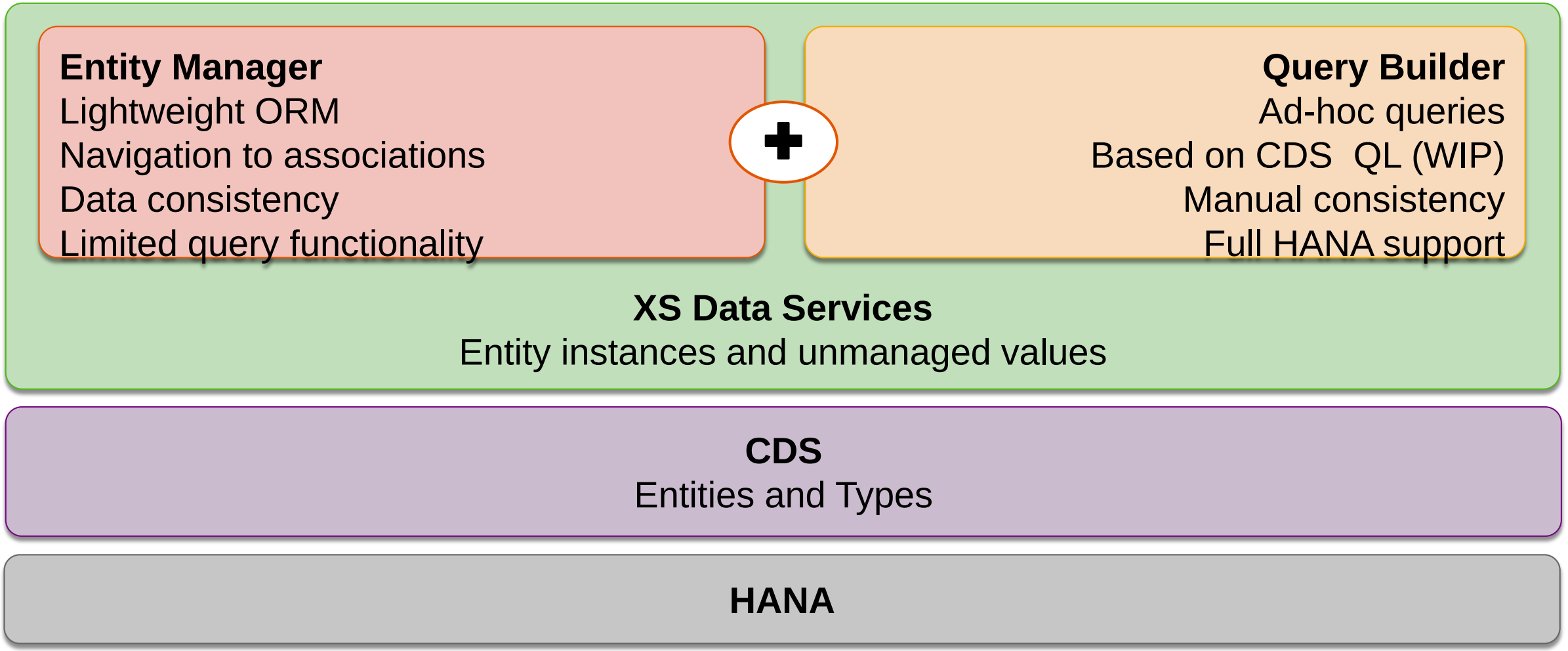
Reads available metadata on types, keys, associations

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)



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.

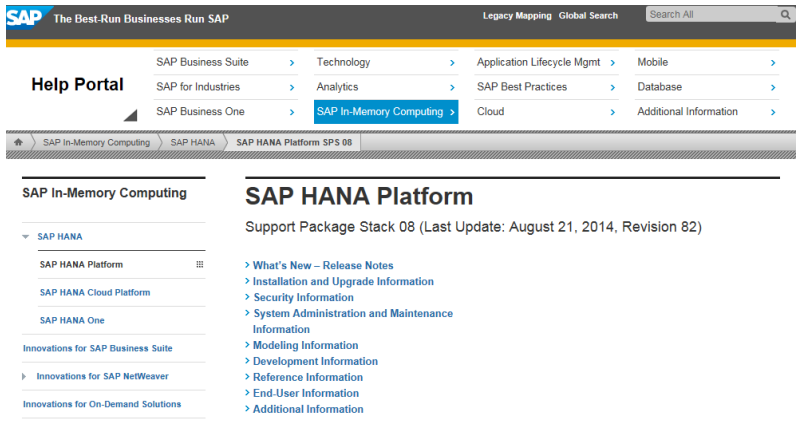
This document is provided without a warranty of any kind, either express or implied, including but not limited to, the implied warranties of merchantability, fitness for a particular purpose, or non-infringement. SAP assumes no responsibility for errors or omissions in this document, except if such damages 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.
- The knowledge centers are structured according to the product lifecycle: installation, security, administration, development:

SAP HANA Platform SPS

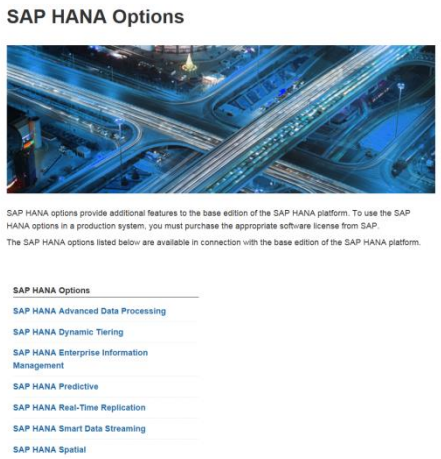
- What's New – Release Notes
- Installation
- Administration
- Development
- References



- Documentation sets for SAP HANA options can be found at 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



Thank you

Contact information

Thomas Jung
SAP HANA Product Management
AskSAPHANA@sap.com

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