Considerations when upgrading SAP Business Workflow from SAP R/3 4.6C to SAP ERP 6.0
Content Providers

Eddie Morris: SAP NetWeaver Tools Support, SAP Ireland
Carol Thomas: SAP NetWeaver Consultant, SAP America
Bob Duffy: SAP NetWeaver Consultant, SAP America
Thomas Kosog: Platinum SAP NetWeaver Consultant, SAP America
Anna Hill: SAP NetWeaver Consultant, SAP UK
Mark Griffiths: SAP NetWeaver Consultant, SAP UK
Jocelyn Dart: SAP NetWeaver Consultant, SAP Australia
Nicole Fessler: Workflow Developer, SAP AG
Markus Kinateder: Senior Developer, SAP AG
Ginger Gatling: NetWeaver Product Manager, SAP Labs
Preparing for the workflow upgrade

During and after the upgrade

New workflow features since R/3 4.6c

Summary and Appendices
Before the upgrade

- Complete all running workflow instances
- Create workflow test plans
- WF & T versus WS & TS
  - Not ‘required’ to switch (Workflow Tasks (WF) and Customer Tasks (T)) to workflow templates (WS) and standard task (TS).
  - Recommend to copy from WF/T to WS/TS.
  - Change can be done after the upgrade (done over time). You can use the transaction PFTC and the copy functionality to do.
Before the upgrade

Clean runtime tables before the upgrade

- If the workflow runtime tables have less data in them then this will enable the upgrade to run faster. The best option is to archive as much workflow runtime data as possible according to note 573656. If you have runtime data that is not needed in the future then you can consider deleting this data via report RSWWWIDE (See note 49545). Please be aware that once deleted that data is no longer available. Various workflow database tables may contain unnecessary entries after you delete work items. It is also a good idea to run report RSWWWIDE_DEP which removes these unnecessary entries.

- 573656 - Collective note relating to Archiving in workflow
- 49545  - Deleting unnecessary work items
- 738148 - Deletion report for superfluous work item entries (46C)
Preparing for the workflow upgrade

During and after the upgrade

New workflow features since R/3 4.6c

Summary and Appendices
During the upgrade

Conversion of event linkage tables

- 1019080 - Poor RSWFEVTXPR A performance
- 808790 - Poor RSWFEVTXPR A performance

In release 4.6C, the instance linkage for workflow steps was stored in table SWEINSTCOU. In WAS (Basis 620) onwards, this information is stored in table SWFDEVINST. During the upgrade, report RSWFEVTXPR A moves the entries from the old table into the new table.

NOTE: If you do the deletion of work items these tables will be cleaned up as well.
After the upgrade

- Composite Upgrade note
  - 1068627 - Composite note about workflow upgrade
  - This note lists the notes concerning the workflow that you must refer to when you upgrade to 6.40 or 7.00. Please make sure all of the notes listed are applied to your system and you have reviewed any recommendations within the notes.
After the upgrade

Workflow Definition & Binding

- Workflow definition
  - 1060762 - Container operation with date fields and time fields (date check routine in workflow builder)
  - 1058159 - Elements are missing in the workflow container (DDIC elements no longer active)
- Binding checks are much more strict in NW04 and 7.0. Binding definitions that did not error or cause problems may now cause problems. Your workflow template containers may refer to DDIC data types that no longer exists in the upgraded system. Note 939489 discusses this issue and provides 2 solutions:
  - Solution 1: The easiest solution is to recreate the missing type definition in the ABAP Dictionary.
  - Solution 2: If Solution 1 is not feasible or useful, you can replace the data types in the container definitions for the relevant processes with a report. Bear in mind that you have to replace them with the same type. Apply note 939489 in order to have the report RSWF_CNT_BOR_ELEM_REPLACE.
- 939489 - Workflow: Container elements are missing after an upgrade
New Binding Editor

source container

drag & drop

import binding

drag & drop

export binding

drag & drop

target container

binding instructions
Binding Expressions

Expressions now provide index access to multi-line elements

<table>
<thead>
<tr>
<th>Expression</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&amp;table[index]&amp;</code></td>
<td>Access to an entire table line</td>
</tr>
<tr>
<td><code>&amp;table[index].columnName&amp;</code></td>
<td>Access to a component of a table line</td>
</tr>
<tr>
<td><code>&amp;table[].columnName&amp;</code></td>
<td>Projection to a single column</td>
</tr>
<tr>
<td><code>&amp;customers[1].orders[2]&amp;</code></td>
<td>Second order of first customer</td>
</tr>
</tbody>
</table>

Expressions can contain functional method calls to BOR or ABAP Objects

- The expression evaluates to the result value of the method
- It is possible to pass parameters
- Methods used in expressions must not have side effects
  - „Read only“ methods
  - No database changes

<table>
<thead>
<tr>
<th>Method Type</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instance method w/o parameters</td>
<td><code>&amp;my_object.getValue( )&amp;</code></td>
</tr>
<tr>
<td>Instance method w parameters</td>
<td><code>&amp;my_object.methodA( param1=&amp;exp1&amp;; param2=17 )&amp;</code></td>
</tr>
<tr>
<td>Static method</td>
<td><code>%my_class.staticMethod( param1=&amp;exp2&amp; )%</code></td>
</tr>
</tbody>
</table>

No more restrictions concerning data types

- ABAP-OO object references
- Strings
- Nested structures
Source and Target of a binding instruction are expressions
- Assigning a table line to a structure component is now possible (if compatible)

Several binding operators are available
- for expression to expression binding (as above)
- for container to container binding

The button lets you choose the right binding operator

<table>
<thead>
<tr>
<th>Expression to expression</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Assign value</td>
<td>Append to table</td>
<td>Initialize</td>
<td>XSLT Transformation</td>
<td></td>
</tr>
<tr>
<td>Container to container</td>
<td>Merge (Copy all elements)</td>
<td>Programmed Binding (Container-IF)</td>
<td>Programmed Binding (SWCONT)</td>
<td></td>
</tr>
</tbody>
</table>

see Appendix for further configuration possibilities
After the upgrade

Tables SWW_CONTOB & SWW_CONT

- Tables SWW_CONTOB & SWW_CONT are used in Basis 640 onwards, but only if you choose the 'old' container persistence. When using XML persistence in your workflow no entries will be written to these tables. You can change the settings for the persistence profile of a workflow via the builder => Basic Data => Version Dependent => Control tab. Look in the 'Persistence profile' tab and you can change the settings.

- Entries with XML persistence are stored in table SWWCNTP0. This should not have any affect on the system at all. Only issue will be if the customer has some custom code that reads from tables SWW_CONTOB & SWW_CONT.
Re-implementation of the Container

Complete reimplementation of container technology

No restrictions concerning data types
  - DDIC types with arbitrary lengths
  - Binary data
  - Structures with arbitrary depth and complexity
  - ABAP OO references

Container is by default stored as XML document
  - Facilitates diversity of supported data types
  - Saves storage space (especially for large containers/structures)
    → No database entry for each structure field required anymore
  - Entails serialisation / deserialization effort

Old container persistence (SWCONT, SWCONTOB) still supported
  - Compatibility mode
  - Workflow Builder setting (Appendix)
  - Old container persistence (SWCONT, SWCONTOB) still supported
Parameter Container Interface

The old container was represented as an internal table

- Manipulating the container meant using awkward container macros

The new container implementation is ABAP Objects based

A (released) API is provided for reading and manipulating container data

- the ABAP Objects interface `IF_SWF_IFS_PARAMETER_CONTAINER` contains the following methods:

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLEAR</td>
<td>Resets an element to its type-based initial value</td>
</tr>
<tr>
<td>GET</td>
<td>Delivers the value of an element</td>
</tr>
<tr>
<td>GET_TYPE</td>
<td>Delivers the data type of an element</td>
</tr>
<tr>
<td>GET_VALUE_REF</td>
<td>Delivers a reference to the value of an element</td>
</tr>
<tr>
<td>LIST_NAMES</td>
<td>Delivers a list of names of all elements</td>
</tr>
<tr>
<td>SET</td>
<td>Sets the value of an element</td>
</tr>
</tbody>
</table>
After the upgrade

**WEB**
- BWSP -> Integrated ITS
- Universal Worklist (Recommended)
- SBWP, inbox in SAPGui

**When moving to Unicode:**
- 775111 - Unicode problem in BOR container
- 1019850 - Contents of WI_Creator field
- 967414 - Error in binding editor with double-byte characters after upgrade
After the upgrade

**SAPGUI**
- If you experience short dumps relating to MESSAGE_TYPE_X it may be due to your SAPGUI Release and patch level. It is a good idea to also upgrade your SAPGUI to the latest release and patch level available on SAP Service Marketplace.

**Deadlines**
- 1092157 - Deadline issues

**Hanging Workflow**
- 215753 - Upgrade: Old workflows hang

**Transports**
- 571302 - Collective note relating to transports in workflow
- 850556 - Workflow: Report for deleting the shared buffer
- 980834 - Buffering of workflow definitions
- 982352 - Updating workflows in the test system (only needed if using both WF and WS tasks)
After the upgrade

Business Object Issues

- If you experience short dumps with DATA_UC_STRUCT_C_LENGTH or ASSIGN_LENGTH_0 then this may be due to the business object(s) being used in your workflow.
  - Solution: Generate the object and any subtypes associated with the object via SWO1
- Transaction SWO_ASYNC
  - After upgrade some users find that they can no longer display the object method by clicking the link in the preview pane of SBWP. This also may affect the users ability to view work item attachments or secondary methods. This is due to missing authorization of transaction SWO_ASYNC which is detailed in note 1006235 - Authorization check for transaction SWO_ASYNC.

Custom code

- RUNTIME HANDLES (TYPE SWC_OBJECT) versus PERSISTENT OBJECT REFERENCES (TYPE SWOTOBJID).

- SAP ERP 6.0 is more stringent in enforcement of ABAP syntax. Example:
  - “We had a macro call which was coded swc_refresh_object "ZECM". I don't know how they ever got away with using the object name in quotes. All of the documentation pointed to defining a field to hold the object values so the call should have been swc_refresh_object self (self being a field holding the value in the correct format)”
New workflow features since R/3 4.6c

Preparing for the workflow upgrade

During and after the upgrade

Summary and Appendices
New workflow features since R/3 4.6c

Design Time Changes and SAP_WAPI at runtime

Delivering work items

Reporting

Note: We only provide overview of changes in this presentation. There is an appendix we will not cover that has more details and there is a delta class, DBITWF.
<table>
<thead>
<tr>
<th>Function group</th>
<th>Function group short text</th>
</tr>
</thead>
<tbody>
<tr>
<td>SWF_API_IO</td>
<td></td>
</tr>
<tr>
<td>SAP_WAPI,message_handler</td>
<td></td>
</tr>
<tr>
<td>SWRA</td>
<td></td>
</tr>
<tr>
<td>SAP_WAPI,ADM,WORKFLOW,CANCEL</td>
<td></td>
</tr>
<tr>
<td>SAP_WAPI,ADM,WORKFLOW,RESTART</td>
<td></td>
</tr>
<tr>
<td>SAP_WAPI,ADM,WORKFLOW,RESUME</td>
<td></td>
</tr>
<tr>
<td>SAP_WAPI,ADM,WORKFLOW,SUSPEND</td>
<td></td>
</tr>
<tr>
<td>SAP_WAPI,ADM,WORKITEM,BACK</td>
<td></td>
</tr>
<tr>
<td>SAP_WAPI,ADM,WORKITEM,REDIRECT</td>
<td></td>
</tr>
<tr>
<td>SWRC</td>
<td>Workflow interfaces: Work list client</td>
</tr>
<tr>
<td>SAP_WAPI,ATTACHMENT,ADD</td>
<td></td>
</tr>
<tr>
<td>SAP_WAPI,ATTACHMENT,ADD,REF</td>
<td></td>
</tr>
<tr>
<td>SAP_WAPI,ATTACHMENT,DELETE</td>
<td></td>
</tr>
<tr>
<td>SAP_WAPI,CHANGE,WORKITEM,PRIOR</td>
<td></td>
</tr>
<tr>
<td>SAP_WAPI,COUNT,WORKITEMS</td>
<td>Workflow interfaces: Number of Work Items for User</td>
</tr>
<tr>
<td>SAP_WAPI,END,RESUBMISSION</td>
<td></td>
</tr>
<tr>
<td>SAP_WAPI,EXECUTE,WORKITEM</td>
<td>Workflow interfaces: Execute Work Item</td>
</tr>
<tr>
<td>SAP_WAPI,FORWARD,WORKITEM</td>
<td>Workflow interfaces: Forward Work Item</td>
</tr>
<tr>
<td>SAP_WAPI,GET,ATTACHMENTS</td>
<td>Workflow interfaces: Read Attachment for Work Item</td>
</tr>
<tr>
<td>SAP_WAPI,GET,DEADLINES</td>
<td>Workflow interfaces: Read Work Item Deadlines</td>
</tr>
<tr>
<td>SAP_WAPI,GET,HEADER</td>
<td>Workflow interfaces: Read Work Item Header</td>
</tr>
<tr>
<td>SAP_WAPI,GET,METHODS</td>
<td>Workflow interfaces: Read Work Item Methods</td>
</tr>
<tr>
<td>SAP_WAPI,GET,MULTI,EXEC,GUID</td>
<td></td>
</tr>
</tbody>
</table>
Step Conditions

Create Work Item Condition

Instantiation

Complete Execution Condition

Complete Work Item Condition

READY

SELECTED

STARTED

CHECKED

Execution

see Appendix for further details

© SAP AG 2007, Upgrading SAP Business Workflow 22
SWITCH: The New Multiple Condition Flavor

1. Double-click on multiple condition icon

2. Select Multiple Condition Flavour

3. Enter Branches and Conditions

4. Name Default Branch

Each branch of the SWITCH carries a condition definition.

The conditions are evaluated in the given order.

The first branch whose condition evaluates to TRUE is taken.

If none of the conditions evaluates to TRUE, a default branch is taken.

Condition 1 is TRUE
Condition 2 is TRUE
Default branch
Blocks...  
- are modeling elements  
- can contain other modeling elements  
- represent a data sub-context  
  - all modeling elements within the block refer to this data context (binding, conditions, ...)
- have a data interface  
- have one start and one end node  
- are represented by block work items  
  - new work item type
- can be deadline monitored (latest end only)
- can catch exceptions (see Appendix)
Blocks: Work Item Hierarchy

- Block items introduce new levels in work item hierarchies
  - Blocks can contain dependent work items

- Blocks are represented as folders in the workflow log

- Block structure is sustained at runtime
  - Block is completed only if dependent work items are in a final state
  - If a block is cancelled, all its dependent work items are cancelled, too
Local workflows...

- are control flow snippets
  - not part of the main process sequence
- are just special blocks
  - represented by block work items
- are started via local events
  - data binding event – local flow
- have full access to process data
- can be instantiated an arbitrary number of times
- are cancelled if still active when main process completes
Local Workflows

1. Exception is raised
2. Exception is caught by block
3. Local event is raised
4. Local workflow is started
Workflow Header Events (1)

- The workflow instance can listen to events.
- Event couplings only exist if the corresponding object exists in the workflow container.
- The receiver type tells you how the workflow processes the event.

<table>
<thead>
<tr>
<th>Receiver Type</th>
<th>Reaction of Workflow to Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cancel Workflow</td>
<td>- The workflow instance is set to status CANCELLED</td>
</tr>
<tr>
<td>Cancel and Restart Workflow</td>
<td>- The workflow instance is set to status CANCELLED</td>
</tr>
<tr>
<td></td>
<td>- A new instance is started with the same data</td>
</tr>
<tr>
<td>Evaluate Pre- and Postconditions</td>
<td>- Conditions „Create Work Item“ and „Complete Work Item“ are evaluated</td>
</tr>
<tr>
<td>Reevaluate Rules</td>
<td>- Agent determination of all active dialog work items is repeated</td>
</tr>
</tbody>
</table>
Workflow Header Events (2)

WAS 640 (NW04) offers you two new receiver types

<table>
<thead>
<tr>
<th>Receiver Type</th>
<th>Reaction of Workflow to Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wait Step Event</td>
<td>Event is delivered to existing appropriate wait step (event work item)</td>
</tr>
<tr>
<td></td>
<td>If no active event work item exists, the event is parked</td>
</tr>
<tr>
<td></td>
<td>➔ see one of the next slides for parked events and the extended wait step!</td>
</tr>
<tr>
<td>Trigger Local Event</td>
<td>A specified local event is triggered</td>
</tr>
<tr>
<td></td>
<td>This local event could</td>
</tr>
<tr>
<td></td>
<td>➢ start local workflows</td>
</tr>
<tr>
<td></td>
<td>➢ complete wait steps (event items)</td>
</tr>
<tr>
<td></td>
<td>➔ see one of the next slides for the extended wait step!</td>
</tr>
</tbody>
</table>
Problem/Feature 1:

- Workflow can only react on events if appropriate wait step has already been instantiated
- If event is raised „too early“, it is lost for the process

Problem/Feature 2:

- Events always complete all corresponding active wait step instances (event items)
Parked Events (2)

The process instance can act as an intermediate event storage

- If active receiver event items exists, the one created first receives the event
- If no active receiver event item exists, the event is parked
- The first matching event item created consumes the first event parked
- In any case: One event is delivered to exactly one event item

See Appendix for further details
Wait Step: Four Different Flavours

Wait for event
- Conventional wait step
- Wait for global event
- Correlations can be used

Wait for event using workflow
- Wait for global event
  - received by workflow instance
  - dispatched to event item
- Correlations can be used

Wait for condition
- Condition specified in wait step
- Evaluated by periodic background job SWWCOND

Wait for local event
- Wait for local event raised by trigger step within the same workflow instance

ASUG
Using ABAP Objects in Workflows (1)

- SAP Business Workflow now supports two object repositories
  - BOR (Business Object Repository)
  - SE24 (ABAP Objects)
- Persistent object keys now have three components
  - Object category (CATID) (BOR or ABAP Objects)
  - Object type (TYPEID) (BOR object type or ABAP Objects classname)
  - Object key (INSTID)
- Both repositories equally well supported
  - But: No delegation feature in SE24
- SE24 support covers the same aspects as BOR support
  - Container
  - Expressions
  - Methods
  - Events
Container can reference ABAP OO objects

ABAP OO objects can be used in expressions
  - just as BOR objects are used

ABAP OO object methods can be used in standard task definitions
  - just as BOR object methods are used
New workflow features since R/3 4.6c

Design Time Changes and SAP_WAPI at runtime
Delivering work items
Reporting

Note: We only provide overview of changes in this presentation. There is an appendix we will not cover that has more details and there is a delta class, DBITWF.
Universal Worklist

- Give users a **unified and centralized access** to their work and the relevant information

- Aggregate task items from **multiple** and different **systems** of the system landscape - in one list for one-stop access
  - Business Workflow
  - Collaboration Task
  - Alert Management System
  - KM Content Management Notifications

- Enable users for direct **decisions** and **actions**

- Support users in **personalization** of the presentation

- Allow extensive **customization** for specialized work lists in Work Centers, including custom attributes, actions and bulk processing
**UWL 7.0 – Task list**

**Work inbox: Tasks in "Flow and in progress tasks (11/27)" view**

- **JDK 1.3/1.4 problem for NW 2004s/7.0 (Downport)**
  - From: Kruse, Matthias
  - Sent: Feb 14, 2006
  - Priority: High
  - Due: Yesterday
  - Status: In Progress

- **Provide XML status**
  - From: Kruse, Matthias
  - Sent: Today
  - Priority: High
  - Due: Tomorrow
  - Status: New

- **Demo NW Team FA**
  - From: Kruse, Matthias
  - Sent: Feb 15, 2005
  - Priority: High
  - Due: Feb 22, 2005
  - Status: New

As you may know parts of NW04 must be build with J2SE 1.3. Upgrading these parts in NW 2004s/7.0 in such a way that they can be build with J2SE 1.4 seems too much effort and would bring too much instability in the NW 2004s codebase. On the other hand Sun does not support that 1.3 compliant Java sources are build with an 1.3 compliant compiler (not even by using cross-compilation) if they have to be compiled against 1.4 class files as indicated in the following picture. Action Item: If your component does currently not comply with the above rule and you are compiling with 1.4, we recommend that you also check immediately how your project is affected by using a local build with J2SE 1.3.
### UWL 7.0 – Launch Work Transaction

<table>
<thead>
<tr>
<th>Subject</th>
<th>From</th>
<th>Sent</th>
<th>Priority</th>
<th>Due</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Notification of absence from: Revise</td>
<td>Workflow System</td>
<td>Feb 15, 2005</td>
<td>Normal 1</td>
<td></td>
<td>New</td>
</tr>
<tr>
<td>Employee CARTMAN: Approve notification of absence</td>
<td>Workflow System</td>
<td>Feb 11, 2005</td>
<td>Normal 1</td>
<td></td>
<td>In Progress</td>
</tr>
</tbody>
</table>

Parameterized launching for:
- BSP
- iView
- URL
- Web Dynpro
- SAP GUI (for HTML)
### UWL 7.0 – Standard Task Views & Filters

#### Tasks (11 / 27)

<table>
<thead>
<tr>
<th>Subject</th>
<th>From</th>
<th>Sent</th>
<th>Priority</th>
<th>Due</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>JDH1 3/M 4 problem for NW2004sp 0 (Downport)</td>
<td>Cartman</td>
<td>Feb 14, 2005</td>
<td>High</td>
<td>Yesterday</td>
<td>In Progress</td>
</tr>
<tr>
<td>Edit maintenance notification 10001257 / M2</td>
<td>Workflow</td>
<td>Aug 26, 2004</td>
<td>Normal</td>
<td>Aug 26, 2004</td>
<td>In Progress</td>
</tr>
<tr>
<td>Edit maintenance notification 10001259 / M1</td>
<td>Workflow</td>
<td>Aug 26, 2004</td>
<td>Normal</td>
<td>Aug 26, 2004</td>
<td>In Progress</td>
</tr>
<tr>
<td>Edit maintenance notification 10001260 / M1</td>
<td>Workflow System</td>
<td>Aug 26, 2004</td>
<td>Normal</td>
<td>Aug 26, 2004</td>
<td>In Progress</td>
</tr>
</tbody>
</table>

#### Show

- New and in progress tasks (11 / 27)
- New and in progress tasks (11 / 27) - select a subview
- All

#### Filters

- Tasks forwarded to someone
- JDH: Tasks for resubmission
- Completed Tasks
- Demo NY Team PA
- Initiate Employee Performance Management
- Notification of absence number 0000002052 rejected. What do you want to do?
- Notification of absence number 0000002092 rejected. What do you want to do?
Launch Work Transaction

Parameterized launching for:
- BSP
- iView
- URL
- Web Dynpro (Java or ABAP)
- SAP GUI (for HTML)
Launch Customization: ABAP Web Dynpro

XML Example

```xml
<Action name="launchWebDynProABAP" handler="SAPWebDynproABAPLauncher"/>
```
Boosting SBWP Performance

Means to improve SBWP performance have been provided

Scenarios that benefit most are

- Many agents work on large pools of work items (e.g. call center scenario)
  - Large, slow work lists
  - Necessity to often refresh the work list (due to work item „overlaps“)

- Extensive usage of dynamic columns
  - particularly with large work lists

- Grouping of work items in large work lists using container data
  - Grouped according to content
  - Grouped according to content type
  - Grouped according to sort key

To mitigate SBWP performance issues, BAdIs have been defined
Email Notifications

Replacement for SAP MAPI is provided
- Client-based MS Outlook integration of SAP Business Workflow
- Support ends 10/2005

New solutions
- Server-based (zero footprint)
- Mail client independent
- Covers 90% of workflow-related MAPI customer scenarios

Different replacements for different SAP releases

Report RSWUWFML2
- Polling report sending email notifications
- Releases 4.6C – 620

Extended Notifications
- Server notification framework
- Releases > 640
Extended Notifications: Overview

SAP Business Workflow
- collect workitems
- send e-mails

Email client
End user
- process workitem

Extended Notifications

SAP GUI or Portal UWL

Email
Extended Notifications: Basic Features

- Notify users about work items that need to be processed
- Send workitems to groupware (e.g. MS Outlook or Lotus Notes) carrying direct callbacks to backend transactions
  - Execution of web-based work items (Web Dynpro, People centric UI, …)
  - Execution of work items via SAP GUI for Windows
New workflow features since R/3 4.6c

Design Time Changes and SAP_WAPI at runtime

Delivering work items

Reporting

Note: We only provide overview of changes in this presentation. There is an appendix we will not cover that has more details and there is a delta class, DBITWF.
The Workflow Information System (WIS) has become obsolete with WAS release 610

Customers do their reporting using BI

Workflow projects with BI requirements
  ♦ could not use standard workflow-related data extraction
  ♦ had to implement their own extraction logic for workflow data

This gap has now been closed
  ♦ Technical work item data is available in BI as infocube
    ➔ How many workflows of type ‘xy’ were executed?
    ➔ What is the average processing time for a given workflow (step)?
    ➔ How many steps of type ‘xy’ were executed by org unit ‘abc’?

Workflow and Business Objects worlds are linked by work item to object relation information available
Further information

- **Practical Workflow for SAP** from SAP Press

- **Classroom Education:**
  - DBITWDF – 2 day workflow delta class,
  - BC401 – 3 days ABAP objects class

- **SDN workflow forum and important blogs**
  - [http://sdn.sap.com](http://sdn.sap.com) Forums -> SAP NetWeaver -> BPM and workflow
  - Why use ABAP OO with workflow?
    - [https://weblogs.sdn.sap.com/pub/wlg/3858](https://weblogs.sdn.sap.com/pub/wlg/3858)
  - Getting started with ABAP OO for workflow
    - [https://weblogs.sdn.sap.com/pub/wlg/3907](https://weblogs.sdn.sap.com/pub/wlg/3907)

- **SDN workflow wiki FAQ maintained by Mike Pokraka (and you)**
Questions?

Q&A
Feedback

Please complete your session evaluation.

Be courteous — deposit your trash, and do not take the handouts for the following session.

Thank You !
- Every block has a container
- Nested blocks form nested data contexts
  - Parent relationships between containers
  - Workflow is the top level block
- Parent context is visible within sub context
- Bindings can be defined from
  - the parent context to the sub context (Import binding at block instantiation)
  - the sub context to the parent context (Export binding after block completion)
Blocks: Dynamic Parallel Instantiation

- Block instantiation can be dynamically controlled by multiline container elements (ParForEach)
  - Similar to the „ParForEach“ dynamic parallel processing for activities!

- For each entry in the multiline element, one block instance is created
  - Corresponding line data transported to the block context via binding

- Process continues after the ParForEach block if either
  - all instantiated parallel blocks are completed or
  - a special join condition evaluates to TRUE after completion of a block
Blocks: Dynamic Sequential Execution

- Block execution can be dynamically controlled by multiline container elements (ForEach)

For each entry in the multiline element, the sequence of steps defined within the block is executed once

- Only one block item is created!
- ForEach loop semantics

- Process continues after the ForEach block if either
  - all table entries are processed or
  - a special condition evaluates to TRUE after completion of a loop sequence

Block definition

Multiline container element

N block sequence cycles
Each Workflow instance has a persistence profile

- It is set in the basic data of the workflow definition
- The default value is
  - "XML Persistence" for new workflow definitions
  - "Compatibility Mode" for legacy workflow definitions
Every binding instruction can be configured

The available configuration options depend on the operator

<table>
<thead>
<tr>
<th>Option</th>
<th>Available for</th>
</tr>
</thead>
<tbody>
<tr>
<td>Handle errors as warnings</td>
<td>All binding operators</td>
</tr>
<tr>
<td>Move-corresponding</td>
<td>Assign, Append</td>
</tr>
<tr>
<td>Overwrite empty target elements only</td>
<td>Assign, Merge, XSLT</td>
</tr>
<tr>
<td>No action if source is initial</td>
<td>Assign, XSLT</td>
</tr>
<tr>
<td>No action if source is empty</td>
<td>Assign, XSLT</td>
</tr>
</tbody>
</table>

The set of binding operators is extensible

You can program your own binding operators

- Use IF_SWF_IFS_BI ND_TRANSFORM for expression to expression operators
- Use IF_SWF_IFS_BI ND_TRANSFORM_CONT for container to container operators
**Step Conditions: Condition Types**

<table>
<thead>
<tr>
<th>Condition Type</th>
<th>Behavior</th>
</tr>
</thead>
</table>
| Create Work Item     | - This condition is evaluated on the surrounding container  
                        - Work item remains in WAITING state until condition evaluates to TRUE  
                        - It can not be executed until condition is fulfilled                              |
| Complete Work Item   | - This condition is evaluated on the surrounding container  
                        - Work item is immediately set to COMPLETED when condition evaluates to TRUE  
                        - Result branch „Complete condition true“ is taken                                |
| Complete Execution   | - This condition is evaluated on the step container itself  
                        - Condition is evaluated after work item execution  
                        - If the condition evaluates to TRUE, the work item is set to COMPLETED  
                        - If the condition evaluates to FALSE, the work item remains IN PROCESS or is even set back to READY (see slide notes for details) |
**Step Conditions: Definition**

1. Double-click on activity icon
2. Select Conditions Tab
3. Select Condition Type
4. Enter condition expression

**Step conditions are**

- formulated on the surrounding container (data context) or the step container
- evaluated either by the workflow engine, a periodic background job (SWWCOND) or event driven
- available for the following step types (with work item representation)
  - Activity
  - User Decision
  - Wait Step
  - Web Activity

**The state of the corresponding work item can be influenced by**

- the fact that a condition is defined
- the result of condition evaluation
Blocks can handle exceptions (TRY-CATCH mechanism)
Exceptions can be raised by a Process Control step (workflow exceptions)
For every workflow exception defined within the process, exception handlers can be defined on every block
A generic (CATCH ALL) exception handler is also available
Exceptions are propagated
→ From their origin up the block hierarchy

Blocks either catch exceptions...
✦ in case appropriate handler is defined

... or propagate them further
✦ in case no appropriate handler defined
✦ or
✦ block already caught exception

If exception is caught
✦ everything inside block is cancelled
✦ exception handler is executed
✦ block is completed normally afterwards

If exception is not caught, process is set to an ERROR state
Event couplings are created using
- event name (design time information)
- event object instance (run time information)

The object instance has to be known to the process

**Problem:** How to wait for events of yet unknown objects?

Knowing an object means knowing its key (⇒ unique identifier)

But: objects can also be identified using semantic information

**Goal:** Wait for events of objects semantically coupled to objects known
Correlations are objects which group other objects by semantic criteria (correlation key)

“objects having to do with each other”

Event couplings can also be created using

- event name (design time information)
- event object type (design time information)
- correlation instance (run time information)

The object instance does not have to be known to the process
  - but at least one correlated object instance has to

Example: know order to identify correlated invoice
Correlations are instantiated using one participating object instance

- (semantic) correlation key has to be determined
  → Example: order ID

Wait step (event item) is instantiated using

- correlation instance (semantic key)
- object type
- event
  → Example: wait for creation of invoice referring to known order

Event of unknown, yet correlated object is delivered to event item via the correlation instance

→ see Appendix for further details
■ When a wait step is instantiated,
  ◆ an event work item is created acting as event listener
  ◆ an event instance coupling (subscription) is written for this event work item

■ Every event triggered completes all active event work items subscribed to it

■ The process (workflow instance) plays no role here
The workflow instance can act as an event dispatcher
   ◆ The workflow instance acts as the event receiver
   ◆ It dispatches those events to its dependent event items

Event work items receive events via the workflow

Only one event work item is completed per event
   ◆ If more than one event work items in question exist, the oldest is taken
   ◆ This mechanism implements an event item queue within the process
If no active event items exist when workflow receives event, the event is *parked*

- Event waits for the next corresponding event item to be created within the process
- This mechanism implements a *queue of parked events*

New event items consume corresponding parked events immediately after their creation
Correlations: Principles (1)

Conventional event couplings:
- Wait steps are instantiated for existing and known object instances
- The object key is used to connect event and wait step (coupling)

Requirement:
- Wait for events of unknown objects
  - Object key is not available at wait step creation time
- Object key cannot be used to link event to wait step

Solution:
- Use semantic coupling between object instances (correlation)
  - One object instance leads to several correlated object instances
- Define wait step not by object key but by **object type + correlation instance**
Correlation instances group object instances by semantic criteria.

- The objects are pulled together by a specially defined correlation key.
  - Those object instances fulfill the correlation instance.
  - They can have different object types.

- A correlation instance can not be created until at least one fulfilling object instance exists.
  - A correlation does not make sense without its objects!
Every correlation instance is based on a correlation definition.

The correlation definition consists of:

- A set of key fields (key definition)
- A set of object types that shall be correlated with each other (correlation components)
- A set of assignment rules between object type data and correlation key fields (one for each correlation component)

For every correlation component, those assignment rules have to fully specify the correlation key!
Correlations: Run Time

Correlations are activated with the help of an object instance

- Object type has to belong to the correlation components

Work items listening to events can use correlations

- If the event is raised by any correlation component, the correlation is checked

Object type B
+ event 1
+ active correlation instance

instantiate wait step
event item with correlation
Extended Notifications: History

Report RSWUWFML
- Polling report
- Uses SMTP/SAPconnect mail connection
- Uses SAP Office’s auto-forward address
- Availability: Release 3.1I

E-Mail Notifications for Business Workflow
- Report RSWUWFML2
  ➞ Successor of RSWUWFML
- WinGui shortcuts instead of R3F attachments
- Availability: Release 4.6C

Extended Notifications for SAP Business Workflow
- Server notification framework
- Availability: WAS 6.40
Notification Concepts: Release Dependencies

- Extended Notifications

- SAP MAPI
- RSWUWFML
- RSWUWFML2

>6.40
6.40
6.20
6.10
4.6
<4.6

© SAP AG 2007, Upgrading SAP Business Workflow 73
Individual mails possible on a work item by work item basis

Or....

The following work item requires processing in your SAP Workflow inbox in system U6D, client 800:

First step in workflow verification

Choose one of the decision options given. This completes the processing of this step.

Before you make a decision, you can display the attachments and objects which have been attached to the user decision. You can also create your own attachments.

If you choose Cancel, the user decision remains in your inbox for processing.

- Execute background step immediately
- Execute background step in a minute

- Display Work Item
- Execute Work Item

If you have problems when logging on to the system, contact your system administrator.
Extended Notifications: Aggregation

Shortcut
Attachments possible

Index of work items

Workitem Description

Generic Decision

Shortcut links

The following new work items require processing in system U8D, client 800:

Overview

First step in workflow verification
First step in workflow verification
First step in workflow verification
Employee Markus Kirchmayr: Approve notification of absence

First step in workflow verification

Choose one of the decision options given. This completes the processing of this step.

Before you make a decision, you can display the attachments and objects which have been attached to the user decision. You can also create your own attachments.

If you choose Cancel, the user decision remains in your inbox for processing.

Generic Decision

Execute background step immediately
Execute background step in a minute

Shortcut links

Display Work Item  Execute Work Item  Back to Overview
The following work item requires processing in your SAP Workflow inbox in system USD, client 500:

First step in workflow verification

Choose one of the decision options given. This completes the processing of this step.

Before you make a decision, you can display the attachments and objects which have been attached to the user decision. You can also create your own attachments.

If you choose **Cancel**, the user decision remains in your inbox for processing.

**Execute background step immediately**  
**Execute background step in a minute**

**Display Work Item**  
**Execute Work Item**

If you have problems when logging on to the system, contact your system administrator.
## Extended Notifications: Feature Comparison

<table>
<thead>
<tr>
<th>Feature</th>
<th>SAP MAPI</th>
<th>RSWUWFML</th>
<th>RSWUWML2</th>
<th>Extended Notifications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deliver workitems to any e-mail client</td>
<td>-</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>SAP Shortcut attachments</td>
<td>-</td>
<td>-</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Prefix AND suffix text (surrounding workitem description)</td>
<td>-</td>
<td>-</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Enhanced address determination</td>
<td>-</td>
<td>-</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Enhanced substitution handling</td>
<td>-</td>
<td>-</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>HTML e-mail (Outlook only)</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>X</td>
</tr>
<tr>
<td>Generic decisions</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>X</td>
</tr>
<tr>
<td>Links to new Web UIs (WebDynpro, etc)</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>X</td>
</tr>
<tr>
<td>Lists of workitems in one e-mail</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>X</td>
</tr>
<tr>
<td>Support workitem forwarding via e-mail</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>X</td>
</tr>
</tbody>
</table>
How To Use ABAP OO Objects in Workflow (1)

Methods needed for persistence handling

<table>
<thead>
<tr>
<th>Method</th>
<th>Level</th>
<th>Visibility</th>
<th>Method type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BI_PERSISTENT</td>
<td>Static</td>
<td>Public</td>
<td></td>
<td>Find Using Local Persistent Object Reference</td>
</tr>
<tr>
<td>FDW_BY_LPOR</td>
<td>Instance</td>
<td>Public</td>
<td></td>
<td>Local Persistent Object Reference</td>
</tr>
<tr>
<td>REFRESH</td>
<td>Instance</td>
<td>Public</td>
<td></td>
<td>Flag to Reload from Database</td>
</tr>
</tbody>
</table>

Standard methods used by runtime

<table>
<thead>
<tr>
<th>Method</th>
<th>Level</th>
<th>Visibility</th>
<th>Method type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEFAULT_ATTRIBUTE_VALUE</td>
<td>Instance</td>
<td>Public</td>
<td></td>
<td>Value of Default &quot;Attributes&quot; (as Data Reference)</td>
</tr>
<tr>
<td>EXECUTE_DEFAULT_METHOD</td>
<td>Instance</td>
<td>Public</td>
<td></td>
<td>Execute Default Methods</td>
</tr>
<tr>
<td>RELEASE</td>
<td>Instance</td>
<td>Public</td>
<td></td>
<td>Release for Garbage Collector to Delete</td>
</tr>
<tr>
<td>CONSTRUCTOR</td>
<td>Instance</td>
<td>Public</td>
<td></td>
<td>Constructor</td>
</tr>
<tr>
<td>CREATE</td>
<td>Static</td>
<td>Public</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CREATE_VIA_API</td>
<td>Static</td>
<td>Public</td>
<td></td>
<td></td>
</tr>
<tr>
<td>APPROVE</td>
<td>Instance</td>
<td>Public</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Application specific methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Level</th>
<th>Visibility</th>
<th>Method type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>APPROVE_AND_SYNCHRON</td>
<td>Instance</td>
<td>Public</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DELETE</td>
<td>Instance</td>
<td>Public</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DISPLAY</td>
<td>Instance</td>
<td>Public</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EXISTENCECHECK</td>
<td>Instance</td>
<td>Public</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UPDATE</td>
<td>Instance</td>
<td>Public</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

ABAP OO classes are built with the class builder (SE24)

ABAP OO classes used in workflows have to implement the interface IF_WORKFLOW

IF_WORKFLOW includes two interfaces

- BI_PERSISTENT (methods for object persistence)
- BI_OBJECT (methods for object runtime)
The methods of `BI_PERSISTENT` implement persistence handling

<table>
<thead>
<tr>
<th>Method</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>FIND_BY_LPOR</td>
<td>▪ Factory method &lt;br&gt;▪ Converts a persistent object reference (POR) to a ABAP OO object instance</td>
</tr>
<tr>
<td>LPOR</td>
<td>▪ Returns the persistent object reference (POR) of the ABAP OO object instance</td>
</tr>
<tr>
<td>REFRESH</td>
<td>▪ Tells the object to reload its state from the database (synchronization)</td>
</tr>
</tbody>
</table>

The methods of `BI_OBJECT` are needed by the workflow runtime

<table>
<thead>
<tr>
<th>Option</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEFAULT_ATTRIBUTE_VALUE</td>
<td>▪ Returns the value of the default attribute used for displaying the object instance</td>
</tr>
<tr>
<td>EXECUTE_DEFAULT_METHOD</td>
<td>▪ Tells the object instance to call its default method &lt;br&gt; ▪ Normally, this is a method which displays the object</td>
</tr>
<tr>
<td>RELEASE</td>
<td>▪ Tells the object instance that it is no longer needed &lt;br&gt; ▪ Possibility for controlled cleanup</td>
</tr>
</tbody>
</table>
All public attributes of an ABAP OO class can be accessed directly within workflow

Method parameters can be of any datatype
- Parameters of BOR object types had to be character based
- XML Persistence profile has to be used for container
All public events defined for the ABAP OO class can be used like BOR object events

- as start events
- as completion events
- as workflow header events
- in wait steps
- in event trigger steps
SBWP-BAdI: Filtering Work Lists

- **BAdI WF_BWP_SELECT_FILTER** enables the reduction of the number of work items displayed in the Business Workplace workflow inbox
  - It imports the entire work list (work item header information)
  - It exports the (reduced) work list to be displayed as inbox

- Custom filtering algorithms can be implemented
  - Example implementation based on random numbers available

- Performance critical operations applied to reduced work list
  - Default attributes, dynamic columns, work item text in different languages, …

- Availability: WAS 620 SP 44, WAS 640 SP 9 (see note 765783)
Dynamic columns can be customized on task level
◆ Display work item container information in Business Workplace inbox

Usage of dynamic columns is performance critical
◆ Read container for every single work item
◆ Instantiate objects and read object attributes
◆ Container and Business Object Repository are not mass-enabled

BAdI WF_BWP_DYN_COLUMN can help to improve performance

The BAdI implementation facilitates
◆ custom buffering
◆ mass selection from database

Availability: Future Support Packages for WAS 620 and WAS 640
◆ see note 848382 (not released yet)
Work item exit classes have to implement the ABAP Objects interface IF_SWF_IFS_WORKITEM_EXIT

Exit classes can be attached to
- Steps (activity, decision, wait, web activity) ➔ corresponding work items
- Blocks ➔ block items
- Workflows ➔ flow items

Exit classes are called at particular times in the work item lifecycle
The workflow runtime provides the work item exit with:

- the name of the lifecycle event
- a work item context object (reference to \IF_WAPI_WORKITEM_CONTEXT\)

The work item context provides methods to:

- read work item data (header, task, agents, texts, ...)
- get container handles (\IF_SWF_IFS_PARAMETER_CONTAINER\)
  - of the work item itself
  - of the super ordinate workflow
- read properties (see next slide)
- write messages to the workflow log
Properties are custom labels for workflow definition entities
- steps (activities, decisions, wait steps, web activities)
- blocks
- workflows

Properties can be used to
- categorize steps
- mark process states (milestones)

Properties are just name value pairs

They can be queried in work item exits (→ custom process reporting)
- „Is the current work item an approval work item?“
- „Is the current process still in negotiation phase?“
Application Specific Workflow Reporting

- Workflow definition
  - workflow started
  - activity
  - step property
    - activity
      - workflow completed
    - activity
      - 1
        - step with work item exit
      - 2
  - workflow completed

- Work item exit
- Reporting tool
- Application specific reporting database
- Work item context
Start options can be specified when testing workflows (SWUS)

- Options enable better error analysis
  - Debugging
  - Tracing

<table>
<thead>
<tr>
<th>Option</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start Workflow Asynchronously</td>
<td>- The workflow instance is started via aRFC</td>
</tr>
<tr>
<td>Simulate background processing</td>
<td>- Background steps are executed synchronously, not via tRFC</td>
</tr>
<tr>
<td></td>
<td>- Valuable for debugging background steps</td>
</tr>
<tr>
<td>Start Dialog Automatically</td>
<td>- Switch off synchronous dialog chains</td>
</tr>
<tr>
<td>Trace Level</td>
<td>- Switch on workflow trace and set trace level</td>
</tr>
<tr>
<td>Debug Mode</td>
<td>- Use aRFC instead of tRFC</td>
</tr>
</tbody>
</table>
Extended start options with release 700

<table>
<thead>
<tr>
<th>Option</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Repeated Starts</td>
<td>Instantiate and start workflow a specified number of times</td>
</tr>
<tr>
<td>Persistence Profile</td>
<td>Specify container persistence profile for single step tasks (TS...)</td>
</tr>
<tr>
<td></td>
<td>The persistence profile for workflows (WS...) is specified in the workflow</td>
</tr>
<tr>
<td></td>
<td>definition</td>
</tr>
</tbody>
</table>
Further Information

Public Web:

www.sap.com
SAP Customer Services Network: www.sap.com/services/

Related SAP Education Training Opportunities

http://www.sap.com/education/