The Workflow Guidebook

Version 1.63 – December 2014

Changes

December 10 2014 – Added information about asyselemtypemenudet/awfelemtypemenudet
October 16 2013 – Added information about BATCHDISTR and BATCHDROPT in awftrans
May 23 2013 – Added information about when awfelemtypemenu is updated
March 22 2013 – Added information about threading and multiple services
December 13 2012 – Added information about awfsubstdetail, awfsubstitute, awfsubstrule, awfsubstruledetail and awfsupervisor
January 18 2012 – Added information about how to debug tasks missing from the approval screen
February 25 2011 – Added information about layout
December 16 2010 – Expanded explanation of how often deadlines are checked
December 13 2010 – Added reference for upcoming ACT named events in 5.6SP1 and description of action_code UI in awftask
October 26 2010 – Removed most references to SP2 because they no longer are relevant, reformatted the regular expressions information
October 22 2010 – Revised edition for Agresso 5.6
October 22 2007 – Added description of how to fix process designer problems to 2.6
August 31 2007 - Typos fixed for next_map and next_id
February 15 2007 - Added 3.3 Debugging

1 Data model

1.1 Element types

55SP2+ has systemtables with corresponding awfelemtypeXX tables for overrides and bespoke work.

1.1.1 Asyselemtype/awfelemtype

Header table of the element type definition.

Has a corresponding table awfelemtype

COLUMN_NAME	VALUE	DESCRIPTION
assembly_name		Assembly name.
attribute_id		Attribute ID.
bflag	1	Action overview enabled for the element type.
	2	Don't aggregate tasks in task list.
	4	Allow no wf state (enables "Leave workflow status unchanged" in the process designer).
	8	E-mail – used by IntellAgent.
	16	From attribute id (element type is based on attribute).
	32	Master file (enables the "Tasks added to Action overview – master file level" in the process designer).
	64	Show in document type properties
	128 (5.5.3+)	Update wf_state as each row in wf_group is finished
	256 (5.5.3+)	Always use unique key when updating wf_state on grouped rows, even if all rows get the same wf_state
	512 (5.6+)	No workflow process can be defined on the element type
class_name		Class Name.
description		Description.
element_type		Unique code to identify the element type.
exception_rule		Workflow treatment if rule does not exist.
	NW	No workflow.
	UI	Mote to items to follow up.
priority		Not in use.

COLUMN_NAME	VALUE	DESCRIPTION
status	N	Active.
	P	Parked.
	C	Closed.
title_no		Title ID.

1.1.2 Asyselemtypedet/awfelemtypedet

Detail table of the element type definition. Lists all tables included in the element type and the structure of them.

Has a corresponding table awfelemtypedet.

COLUMN_NAME	VALUE	DESCRIPTION
attribute_id		Attribute ID. Used if a table contains various attributes, e.g agldimvalue.
bflag	1	Not in use.
	2	Exclude from check sum.
	4	Table may not always contain rows for a given item.
column_name		Not in use.
databag_id		Not in use.
element_type		Unique code to identify the element type.
entity_assembly		Entity assembly for table.
entity_def		Entity class definition for table.
hist_table		Historical table.
hist_table_det		Historical detail table.
hist_table2		Historical table 2 (currently not in use).
hist_table2_det		Historical detail table 2 (currently not in use).
query		Not in use.
real_table		Name of real table when using a view in element type. The table that contains the wf_state.
sequence_no		Sequence number.
table_det		Name of detail table, this is blank for the lowest level table in the structure. This table must have column wf_state.
table_name		The table name.

COLUMN_NAME	VALUE	DESCRIPTION
title_no		Standard title ID.
treat_code		Treat code. Used if a table contains various transaction types, e.g acrtrans.

1.1.3 Awfelemtypeitem

Contains the columns chosen in element type details (WF19).

COLUMN_NAME	VALUE	DESCRIPTION
att_id_col		Column name containing attribute id.
attribute_id		Attribute ID.
bflag	1	Trigger redistribute (New distribution in WF19).
	2	Allow in delay setup.
	4	Allow in rules.
	8	Show in task list.
	16	Show in enquiry.
	32	AO Selection list.
	64	Show in DS mapping.
	128	Log value.
	256	IntellAgent e-mail.
	512 (5.6+)	Master file approval
column_name		Column in WF19.
data_item_no		Unique identifier.
data_length		Character length of attribute values.#Values: 1-99.
data_type		Data type of column, same values as found in asyscolumns:
	8	Integer (64 bit.)
	b	Bool.
	c	Fixed length string.
	d	Datetime.
	f	Floating point number.
	g	Guid.
	i	Integer (32 bit).
	l	Long_text/blob.
	m	Money (float).
	r	Blob.
	t	Text.
element_type		Element type.
field_op		Operator in WF19 (used for

COLUMN_NAME	VALUE	DESCRIPTION
		trigger redistribute).
sequence_no		Sequence number (used for show in tasklist, there can only be one of each value from 1-5).
sequence_no2		Sequence number 2 (used to determine the order of columns chosen for show in enquiry).
sequence_no3		Sequence number 3 (used to control the order of columns in action overview).
table_name		Table name in WF19.
title_no		Standard title ID.

1.1.4 Awfelemtypegrouping

Holds the aggregation definition for an element type, defined in Aggregate (WF26).

COLUMN_NAME	DESCRIPTION
attribute_id	Attribute in WF26 (tip: to aggregate all values for an item use A3 - CLIENT).
element_type	Element type.
sequence_no	SequenceNo in WF26.
type	Grouping type. Values: A - Aggregation in rule.

1.1.5 Asyselemtypemenu/awfelemtypemenu

Holds the available screens and server process for an element type, used in the process designer. Rows are added in awfelemtypemenu by Agresso in two cases:

1) A new element type is made in Element types (WF16) based on a master file attribute
2) A new function is defined in Master file approval setup (TWF020)

COLUMN_NAME	VALUE	DESCRIPTION
element_type		Unique code to identify the element type.
entry_flag	0	Screen is not entry screen.
	1	Screen is entry screen.
menu_id_office		The menu id of the screen in Smart client. This is the screen that will be opened when clicking in the task list.
menu_id_web		The menu id of the screen in Self Service. This is the screen that will be opened when clicking in the task list.
menu_ref		Unique id, referenced in awfprocfunction.

COLUMN_NAME	VALUE	DESCRIPTION
s_usage		Usage: • WFS – Screen (user step), • WFR – Report (system step), • REG – registration screen for document type, • BATCH – server process for document type, • AO – action overview screen.
title_no		Standard title ID.

1.1.6 Asyselemtypemap/awfelemtypemap

Holds mapping information between element type and registration screen/server process. Used when loading documents to document archive and sending to registration screen for data entry, e.g. scanning invoices to enter in VP10.

COLUMN_NAME	DESCRIPTION
data_item_no	Unique identifier.
element_type	Element type.
field_name	Name of the field in the screen.
func_name	Name of the screen: Link to asysfunction. Name of the screen or report VP10, EI02 etc.
func_type	Menu function type. Link to asysfunction.
populate_seq_no	The sequence to populate the fields in. More than one field may have the same populate_seq_no.

1.1.7 Asyswfblmethods/awfblmethods

Asyswfblmethods contains definitions of system functions used in OR splits.

Awfblmethods contains ACT implementations of OR split functions and system steps. Please check the appendix section on ACT named events for how these can be implemented.

COLUMN_NAME	VALUE	DESCRIPTION
Assembly_name		Assembly name.
Bflag		Not used
Class_name		Class Name.
Data_type		The data type returned by the function
	b	Bool
	s	String
	d	Double
	i	Int
Element_type		The element type that can use the OR-split or system step
Method_id		Unique key, must be unique across asyswfblmethods/awfblmethods
Method_name		The name of the function
Method_type		Not used
Title_no		The title of the function displayed in the process designer

1.1.8 Asyselemtypemenudet/awfelemtypemenudet

Exists in Milestone 4+
Holds alternate menu_ids for a menu_ref defined in asyselemtypemenu/awfelemtypemenu.

The system will use the menu_id with the highest priority value that the current user has access to for a menu_ref/tree_type combo. If the user does not have access to any of the menu_ids defined in asyselemtypemenudet/awfelemtypemenudet, the value from menu_id_office or menu_id_web in asyselemtypemenu/awfelemtypemenu will be used.

COLUMN_NAME	VALUE	DESCRIPTION
menu_id		The menu id of the screen/system step.
menu_ref		Part of the unique key, referenced in asyselemtypemenu/awfelemtypemenu.
tree_type		Part of the unique key. Defines which environment the menu_id is defined for.
	1	Desktop
	2	Web
Priority		Part of the unique key. The system will use the row with the highest priority value where the user has access to the menu_id.

1.2 Process definition

1.2.1 Awfversion

Determines which version of a process is active.

COLUMN_NAME	DESCRIPTION
Active_flag	Set to 1 for the active version.
Date_from	When the version was set active.
Date_to	When the version was superseded by a newer version.
Description	Comment when the version was committed.
Node_id	Process node_id.
Version_no	The process version.

1.2.2 Awfprocess

The process tree, processes and steps are stored here.

COLUMN_NAME	VALUE	DESCRIPTION
Action_type		Not in use on this table.
Allow_change		“Users allowed to change data” in the Other tab on a user step.
Allow_single		

COLUMN_NAME	VALUE	DESCRIPTION
Auth_type		Not in use.
Auto_app		“Automatic approval” in the Other tab on a user step.
Availability		“Set available for manual use” in general tab on user step.
Bflag	1	Advanced mode is default.
	2	Disable simple mode (implies bit 1 is on)
	4	Two step approval
	8	Master file approval process
	16	Do not include manual approvers when checking two step approval
Condition_id		Not in use.
Description		Description on step boxes in process designer, also shown in general tab.
Do_single_check		“Perform single user involvement…” in other tab on user step.
First_flag		Set to 1 for the first step in a process, as well as the topmost item on each branch in the process tree. **If no nodes have this flag set for a process or a branch in the process tree then nothing gets drawn in the process designer.**
Fraud_detection		Not in use (is always turned on).
Func_name		Not in use.
Func_type		Not in use.
Head_app		“Allow processing on header level” in other tab on user steps Instructions – “Step instructions” in instructions tab on user steps.
Limit_check		Not in use.
Limit_check2		Not in use.
Log		“Tasks added to Action overview – transaction level” on general tab for user steps.
Log_master		“Tasks added to Action overview – master file level” on general tab for user steps.

COLUMN_NAME	VALUE	DESCRIPTION
Long_info		"Description" in general tab for system and user steps.
Next_id		Points to the next node_id in the process tree or process:
	-1	The end.
	-2	Split node, multiple next nodes found in target_id in awfprocsplit.
Next_join_id		Not in use.
Node_id		Identifies a node (either an item in the process tree, or a step in a process).
Node_type	0	Folder.
	1	Process.
	2	Not in use.
	3	User step.
	4	System step.
	5	Delay.
	6	Or split
	7	And split.
	8	Or join (not shown in process designer).
	9	And join.
	10	Sync step.
	11	Subprocess.
No_wfstate		"Leave workflow status unchanged" on general tab for processes, only allowed to change for element types with bflag 4 (currently only RES).
Parent_id		Points to the parent node in the process tree, points to the process for steps in a process.
Previous_id		Points to the previous node_id in the process tree or process:
	-1	No previous node.
	-2	Multiple nodes point to this, check which nodes have next_id equal to this node's node_id.
Prev_join_id		Not in use
Rule_group_id		"Connect to rule group" on general tab in user step.
Show_all		"Display entire transaction" on other tab in user step.
Simpl_rules		Not in use.

COLUMN_NAME	VALUE	DESCRIPTION
Single_check		"Include in single user involvement check" on other tab in user step.
Status	N	Normal.
	P	Parked process on general tab for process.
Stop_distr		"Pause distribution after this step" on other tab in user step.
Subprocess		"Use as sub-process only" on general tab for process. **If this is selected no distribution will be generated for this process unless it is used as a sub process.**
Target_id		Not in use.
Treatment1		"Treatment if change triggers new rule" on other tab in user step.
	NND	No new distribution.
	RED	Redistribute within step.
	FRD	Full redistribution.
	(blank)	Treated as redistribute within step.
Treatment2		Not in use.
Validation_code		"User validation on save" on other tab in user step.
	P	Enter password.
Variant		Not in use.
Version_no		The process version.

1.2.3 Awfprocsplit

Used by OR/AND splits to branch processes.

COLUMN_NAME	VALUE	DESCRIPTION
Action_code		"Action" in OR-split setup tab for OR split branches.
Bflag	1	Unused.
	2	Unused.
	4	"Else" in OR-split setup tab for OR split branches.
	8	"Rollback" in OR-split setup tab for OR split branches.
Description		"Description" in OR-split setup tab for OR split branches.

COLUMN_NAME	VALUE	DESCRIPTION
Element_type		"Element type" in OR-split setup tab for OR split branches.
Node_id		Identifies the split node.
Rule_id		The rule that must match to take this OR split branch.
Sequence_no		Used to uniquely identify a branch.
Target_distr		"Recipient" in OR-split setup tab for OR split branches:
	R	Rule.
	I	Workflow initiator.
	L	Last owner of task.
Target_id		The node_id that this branch points to.
Version_no		Identifies process version.

1.2.4 Awfprocelemtype

Connects element_types to a process.

COLUMN_NAME	DESCRIPTION
Element_type	The element type.
Node_id	The process node_id.
Version_no	The process version.

.

1.2.5 Awfprocfunction

Connects functions (screens or reports) to user and system steps.

COLUMN_NAME	DESCRIPTION
Action_type	"Action type" on general tab for user steps.
Element_type	"Element type" on general tab for user and system steps.
Menu_ref	"Function" on general tab for user and system steps. Points to the menu_ref in asys/wfelemtypemenu.
Menu_ref_ao	"Action overview function" on general tab for user and system steps. Points to the menu_ref in asys/wfelemtypemenu.
Node_ id	The step that this function is connected to.
Variant	"Variant" on general tab for user and system steps.
Version_no	The process version.

1.2.6 Awfprocaction

Connects actions to a user step.

COLUMN_NAME	DESCRIPTION
Action_code	The action code.
Description	Not in use.
Node_id	The step that this action is connected to.
Sequence_no	Controls the sequence of actions.
Version_no	The process version.

1.2.7 Awfprocdelay

Stores delay step properties.

COLUMN_NAME	VALUE	DESCRIPTION
Bflag	1	Set the first time a delay node with new_flag set is run.
Distr_type		Not in use.
Element_type		"Element type" on Delay tab for delay steps.
End_time		Not in use.
Event_id		Not in use.
Last_run		When the delay node last was checked by the workflow service.
New_flag		"New items only" on Delay tab for delay steps.
Node_id		The delay step node_id.
Number_1		"Number" on Delay tab for delay steps.
Run_interval		"Minutes" on Delay tab for delay steps
Start_time		Not in use.
Time_type		Not in use.
Time_unit		"Time unit" on Delay tab for delay steps.
	D	Days.
	H	Hours.
	M	Minutes.
	W	Weeks.
Version_no		The process version.

1.2.8 Awfprocdelaydet

Stores rows in the spread in delay step properties.

COLUMN_NAME	VALUE	DESCRIPTION
Attribute_id		"Attribute" on Delay tab for delay step.
Column_name		"Column name" on Delay tab for delay step.
Column_name2		"Based on" on Delay tab for delay step.
Delay_type		"Type" on Delay tab for delay step.
	1	Time.
	2	Column.
	3	Attribute/relation.
Element_type		"Element type" on Delay tab for delay steps.
Field_no		Not in use.
Field_op		Not in use.
Node_id		The delay step node_id.
Operator1		"Operator" on Delay tab for delay steps.
Rel_attr_id		"Relation" on Delay tab for delay steps.
Sequence_no		Uniquely identifies the row within the delay.
Table_name		"Column name" or "Based on" on Delay tab for delay step.
Time_type		"Time type" on Delay tab for delay step.
	A	After.
	B	Before.
Time_unit		"Time unit" on Delay tab for delay step.
	D	Days.
	H	Hours.
Value		"Value" on Delay tab for delay step.
Version_no		The process version.

1.2.9 Awfprocdeadlines

The deadlines connected to a user step.

COLUMN_NAME	VALUE	DESCRIPTION
Action_code		"Perform" on Deadlines tab for user steps.
	RM	Reminder.
	AP	Approve.
	ES	Escalate.
		User defined action codes.
Column_name		"Based on" on Deadlines tab for user steps.
Field_no		Not in use.
Include_parked		"P" on Deadlines tab for user steps.
Involve		"Involve" on Deadlines tab for user steps
	USR	User.
	SUP	Supervisor.
Last_run		When the deadline last was checked.
Node_id		The step the deadline applies to.
Number_1		"Number" on Deadlines tab for user steps.
Schedule_id		Schedule_id for scheduled deadlines
Time_type		"Time type" on Deadlines tab for user steps.
	A	After.
	B	Before.
	R	Running.
	S	Scheduled
Time_unit		"Time unit" on Deadlines tab for user steps.
	D	Days.
	H	Hours.
	M	Minutes.
	W	Weeks.
Type		"Type" on Deadlines tab for user steps.
	STEP	Step.
	TASK	Task.
Version_no		The process version.

1.2.10 Awfprocrecipient

Users and roles that have been stored in the Recipients tab for a user step.

COLUMN_NAME	VALUE	DESCRIPTION
Bflag	1	IsUser - Defines if the entry denotes a user or a role
	2	Manual Distribution - The entry is present in MD restriction list
	4	FYI - The entry is present in For Your Information (Inform) restriction list
	8	Forward - The entry is present in Forward restriction list
Node_id		The node_id.
Recipient		A role or user tied to the user step.
Version_no		The process version.

1.2.11 Awfuseraction

Contains user defined actions, defined in User defined action (WF22).

COLUMN_NAME	VALUE	DESCRIPTION
Action_code		Action code in WF22 (must be at least three characters).
Availability		General in WF22.
Bflag	1	Allow this action to be used as an automatic action in deadlines
Description		Description in WF22.
Description2		Description of the action in the past tense, used when displaying comments
Element_type		Element type in WF22.
Node_id		Not in use.
Status		Not in use.
Sys_act_code		Behaviour in WF22.
	AP	Positive.
	RJ	Negative.

1.2.12 Awfmssetup

Contains header information for generic master file approval screens

COLUMN_NAME	VALUE	DESCRIPTION
Description		The name of the master file approval screen
Element_type		The element type
Menu_ref		Used as a foreign key in awfprocfunction to identify

		the approval screen for a given step
Sequence_no		Used to identify the correct approval screen when there is more than one for a given element type
Status	N	Active
	P	Parked
Sys_act_code		Behaviour in WF22.
	AP	Positive.
	RJ	Negative.

1.2.13 Awfmssetupdet

Contains information about the fields displayed in a generic master file approval screens

COLUMN_NAME	VALUE	DESCRIPTION
Bflag		Not in use
Column_name		The column name
Detail	1	This column is displayed as a field in the header information on the approval screen
Element_type		The element type
Header	1	This column is displayed as a field in the pane on the left of the approval screen
Sequence_no		Identifies which approval screen the row is connected to
Sequence_no2		Used to order header fields
Sequence_no3		Used to order detail fields
Table_name		The table name

1.2.14 Acrdiagramlayout

Maps between a version of a process and the layout stored in aimblob

COLUMN_NAME	VALUE	DESCRIPTION
Blob_id		A reference to the blob containing the layout in aimblob
Client		The client
Diagram_id	“node_id;version_no”	The node id and version of the process that the layout is stored for
Diagram_type	WF_SUBPROCESS	The type of layout stored, only process layouts exist here for now

1.2.15 Aimblob

Used to store the layout of the workflow process (is also used to store many other things)

COLUMN_NAME	VALUE	DESCRIPTION
Blob_id		Primary key
Blob_image		Binary data
Blob_size		The size of the blob
File_name		Blank for workflow process layouts
Interface	LAYOUT	For rows containing process layouts

1.3 Rules

1.3.1 Awfrulegroup

This is where we store our rule groups.

COLUMN_NAME	VALUE	DESCRIPTION
Description		"Rule group" in WF09.
Group_type		Is this a distribution rule or an OR-split rule.
	D	Distribution.
	O	OR split (not displayed in WF09, configured on OR split branches).
Rule_group_id		Uniquely identifies the rule group.
Status		Topmost "Status" in WF09.
	N	Normal
	C	Closed
Version_no		The process version, only used for OR split rule groups.

1.3.2 Awfrule

Header properties for rules are stored here.

COLUMN_NAME	VALUE	DESCRIPTION
Date_from		"Date from" in WF09.
Date_to		"Date to" in WF09.
Description		"Rule" in WF09.
Field_no		"Rule based on" in WF09, these values are retrieved from awffield.
Group_type		No longer used.
Priority		"Priority" in WF09.
Rule_group_id		The rule group this rule belongs to.
Rule_id		The rule_id that uniquely identifies this rule.
Status		The bottommost "Status" in WF09.
	N	Normal.
	C	Closed.
Version_no		The process version, only used for OR split rules.

1.3.3 Awfruledet

Data specification and routing for rules.

COLUMN_NAME	VALUE	DESCRIPTION
Att_parent_no		Not used.
Attribute_id		"Attribute" in WF09.
Att_val_from		Not in use.
Att_val_op		Topmost "Operator" in WF09.
	=	Like.
	!=	Not like.
	<>	Between.
	!<>	Not between.
	>	Greater than.
	>=	Greater than or equal to.
	<	Less than.
	<=	Less than or equal to.
	()	In list.
	!()	Not in list.
	.=	Like regular expression
	!.=	Not like regular expression
Att_val_to		Not in use.
Att_value		Not in use.
Column_name		"Column name" in WF09.

COLUMN_NAME	VALUE	DESCRIPTION
Element_type		"Element type" in WF09.
Field_op		Bottommost "Operator" in WF09.
Number_from		Not in use.
Number_to		Not in use.
Priority		"Seq" in WF09.
Recip_type		Not in use.
Recipient		"Recipient" in WF09.
Recipient_group		Not in use.
Rel_attr_id		"Relation" or "Via relation" in WF09.
Rel_val_from		Not in use.
Rel_val_to		Not in use.
Rel_val_op		"Operator" in WF09.
Rel_value		"Value" in WF09.
Rule_id		Uniquely identifies a rule.
Rule_type		Bflag.
	1	Routing.
	2	Not in use.
	4	Attribute.
	8	Relation.
	16	Not in use.
	32	From function recipient.
	64	From flexi field group recipient.
	128	Not in use.
	256	Not in use.
	512	Not in use.
	1024	Element type.
	2048	From list recipient.
	4096	Roles from list recipient.
	8192	From attribute recipient.
	16384	From column recipient.
	32768	Recipient from element type (SP2).
	65536	"Aggregate" in OR-split setup tab (requires patch on SP1).
Sequence_no		Uniquely identifies a detail row within a rule.
String_from		Contains the value used in the rule.
String_to		Not in use.
Table_name		Table name for element type rules.
Value_from		Not in use.
Value_to		Not in use.

COLUMN_NAME	VALUE	DESCRIPTION
Version_no		Process version for OR split rules.

1.3.4 Asyswffield

Generic column mapping, used in “Rule based on” in WF09.

COLUMN_NAME	DESCRIPTION
Field_no	Uniquely identifies the field.
Title_no	The column title.

1.3.5 Awfcolumns

Maps generic columns to specific columns.

COLUMN_NAME	DESCRIPTION
Field_no	Uniquely identifies the field.
Wf_column_name	The column to map the generic column to.
Wf_table_name	Table name, can be blank.

1.4 User information

1.4.1 Awfalternate

Contains definition of supervisors and substitutes, defined in the Alternates tab of Workflow user information (WF15). In use until 5.6.1, then replaced by awfsubstdetail, awfsubstitute, awfsubstrule, awfsubstruledetail and awfsupervisor

COLUMN_NAME	VALUE	DESCRIPTION
Act_date_from		Active from date in WF15, used only for substitutes.
Act_date_to		Active to date in WF15, used only for substitutes.
Alternate_type		The type of alternate.
	SUP	Supervisor.
	SUB	Substitute.
Attribute_id		Attribute in WF15.
Bflag		Find alternate in WF15, determines how the recipient is found.
	4	Attribute value recipient.
	8	Via relation recipient.
	2048	Recipient chosen in list.
Date_from		Date from in WF15.
Date_to		Date to in WF15.
Recipient		Alternate in WF15.
Recurrence_id		Not in use.
Rel_attr_id		Via relation in WF15.
Role_id		Role in WF15.
Sequence_no		Used in unique key.
Wf_user_id		User in WF15.

1.4.2 Awfuserdetail

Contains definition of amount limits, defined in the User detail tab of Workflow user information (WF15).

COLUMN_NAME	VALUE	DESCRIPTION
Amount_limit	Limit	Initiated in WF15 (how much you can approve on items you have entered yourself).
Amount_limit2	Limit	Allocated costs in WF15 (how much you can approve when your resource_id or user_id is in the accounting string).
Element_type		Element type in WF15.
Mail_flag		Not in use.
Role_id		Role in WF15.

COLUMN_NAME	VALUE	DESCRIPTION
WF_user_id		User in WF15.

1.4.3 Awfsupervisor

Contains definition of supervisors, defined in the Supervisor setup tab of Workflow user information (WF15). In use from 5.6.2

COLUMN_NAME	VALUE	DESCRIPTION
Attribute_id		Attribute in WF15.
Bflag		Find alternate in WF15, determines how the recipient is found.
	4	Attribute value recipient.
	8	Via relation recipient.
	2048	Recipient chosen in list.
Date_from		Date from in WF15.
Date_to		Date to in WF15.
Element_type		The element type the supervisor applies for
Recipient		Supervisor in WF15.
Rel_attr_id		Via relation in WF15.
Role_id		Role in WF15.
Sequence_no		Used in unique key.
Wf_user_id		User in WF15.

1.4.4 Awfsubstitute

Contains header information about the substitute status of the user, defined in “My substitutes” and “Assign substitutes” (TWF012). In use from 5.6.2

COLUMN_NAME	VALUE	DESCRIPTION
Absent		Absence status in TWF012.
	1	Out of office
	2	In the office
Act_date_from		Absence date from in TWF012.
Act_date_to		Absence date to in TWF012.
Wf_user_id		WF user in TWF012.

1.4.5 Awfsubstdetail

Contains information about assigned substitutes for a user, defined in “My substitutes” and “Assign substitutes” (TWF012). In use from 5.6.2

COLUMN_NAME	VALUE	DESCRIPTION
Date_from		Valid from in TWF012.
Date_to		Valid until in TWF012.
Element_type		The element type the substitute applies to

COLUMN_NAME	VALUE	DESCRIPTION
Recipient		Substitute in TWF012
Sequence_no		Used in unique key
Type		Type in TWF012
	G	Generic substitute
	E	Substitute only for the given element type
Wf_user_id		WF user in TWF012.

1.4.6 Awfsubstrule

Contains header information for the setup of possible substitutes for a user, defined in "Administer workflow substitutes" (TWF026). In use from 5.6.2

COLUMN_NAME	VALUE	DESCRIPTION
Attribute_id		Attribute ID in TWF026. Can be C0 (resno) or GN (user)
Bflag		Find substitute in TWF026.
	4	Relation on attribute
	8	Via another relation
	2048	From list
Recipient		Recipient in TWF026. Can be C0 (resno) or GN (user)
Rel_attr_id		Via relation in TWF026.
Sequence_no		Used in unique key
Wf_user_id		WF user in TWF026.

1.4.7 Awfsubstruledetail

Contains detail information for the setup of possible substitutes for a user, defined in "Administer workflow substitutes" (TWF026). In use from 5.6.2

COLUMN_NAME	VALUE	DESCRIPTION
Exclude_user		Substitutes to be excluded after the corresponding rule in awfsubstrule is applied (bflags 4 and 8)
Include_user		Substitute which has been selected from list (bflag 2048)
Sequence_no		Used in unique key
Wf_user_id		WF user in TWF026.

1.5 Workflow map

1.5.1 Awfobjectmap

Connection between workflow and items on workflow.

COLUMN_NAME	VALUE	DESCRIPTION
Bflag (5.6)	1	Row has been submitted with wf_state X, Y or Z
Check_sum		Used to check for changes in the item on workflow.
Composite_key		For debug purposes.
Oid		Uniquely identifies an item on workflow.
Map_table_name		The map table that contains the workflow item key columns.
Wf_group		Groups different items on workflow (i.e. rows in an invoice).

1.5.2 Awfobjectlog

Comments and information about happens to an item

COLUMN_NAME	VALUE	DESCRIPTION
Argument		Argument used to fill in parameterized titles in title_no
Bflag	1	First log for row
Comments		The comment entered for a row
Element_type		The element type for a row
Log_id		The unique key for a log item
Log_type	CHC	Change mapping columns
	DEA	Deadline action
	ENT	Entered
	EXT	External
	MIS	Miscellaneous
	REM	Reminder sent
	SPL	Row splitted
	FYI	FYI task created
	PARK	Task parked
Map_id		Identifies which awfmap row the log item is connected to, if any
Oid		Uniquely identifies an item on workflow.
Task_id		Identifies which awftask row the log item is connected to, if any
Title_no		The log item message
Wf_group		Groups different items on workflow (i.e. rows in an

		invoice).

1.5.3 Awfmaphead

Corresponds to the process level in awfprocess/process definition.

COLUMN_NAME	VALUE	DESCRIPTION
Action_code		The last action in the workflow map.
Bflag	1	Sub process.
	2	Comments in map.
Distr_date		when the map was distributed.
Element_type		Element type of the item.
Finished_date		When the map was finished.
Initiator		The user that distributed the workflow map.
Map_head		Uniquely identifies a workflow map.
Node_id		The node_id of the process node.
Oid		Uniquely identifies an item on workflow.
Version_no		The version of the process node.
Wf_group		Groups different items on workflow (i.e. rows in an invoice).
Wf_status	ABO	Aborted.
	ACT	Active.
	DEC	Deactivated.
	FIN	Finished.
	FNS	Finished (sub process).
	TMD	Timed out.
	WTN	Waiting.

1.5.4 Awfmap

Corresponds to a step in awfprocess/process definition

COLUMN_NAME	VALUE	DESCRIPTION
Bflag	1	Changes in map.
	2	Delay triggered before map active (No longer used.).
	4	Speculative distribution.
	8	Redistributed.
	16	Comments in map.
Distr_date		When the map was distributed.

COLUMN_NAME	VALUE	DESCRIPTION
Distr_user		The user that distributed the map.
Element_type		Element type of the item.
Finished_date		When the map was finished.
Last_deadline		When the last deadline triggered for the map.
Logged_values		Stores values chosen for logging in element type details.
Long_info		Stores distributor comments.
Map_group		Used together with next_map to tie together the steps within the map_head.
Map_head		Uniquely identifies an instance of a process.
Map_id		Uniquely identifies an instance of a step in a process.
Next_map		Points to the next map_group within the map_head.
	-1	No more maps.
	-2	Paused distribution.
Node_id		The node_id of the step.
Oid		Uniquely identifies an item on workflow.
Ready_date		When the map was made active.
Step_type	A	And split.
	D	Delay.
	G	Sub process.
	M	Manual user step.
	N	And join.
	O	Or split.
	P	User step.
	R	Or join.
	S	System step.
	Y	Sync step.
Version_no		The version of the step.
Wf_group		Groups different items on workflow (i.e. rows in an invoice).
Wf_status	ABO	Aborted.
	ACT	Active.
	DEC	Deactivated.
	FIN	Finished.
	SKP	Skipped.
	TMD	Timed out.

COLUMN_NAME	VALUE	DESCRIPTION
	TRG	Triggered delay (no longer used).
	WTN	Waiting.

1.5.5 Awftask

Corresponds to a task for a user within a step instance.

COLUMN_NAME	VALUE	DESCRIPTION
Action_code		The action taken on the task, two letter codes are system defined actions, others are defined in **awfuseraction.**
	AT	Auto approved.
	AB	Aborted.
	AP	Approved.
	ES	Escalated.
	FW	Forwarded.
	PA	Parked, will be replaced by a real action code when the item no longer is parked.
	RJ	Rejected.
	SP	Split, this action code will only be seen in awftask for 5.5 SP1. From 5.5 SP2 onwards a real action code must be set on split rows.
	UI	Processed in items to follow up, should only be seen on tasks with distr_type U
Action_date		When the action was taken.
Active	1	If the task still is valid.
Bflag	1	First task in map.
	2	Comments in map.
Col?_descr		Description for a column chosen to be displayed in the task list in element type details.
Col?_dt		Data type for a column chosen to be displayed in the task list in element type details.
Col?_value		Value for a column chosen to be displayed in the task list in element type details.
Comments		Comment from when task was performed.
Distr_date		When the task was distributed.

COLUMN_NAME	VALUE	DESCRIPTION
Distr_type		How the task was generated.
	E	Escalated.
	F	Forwarded.
	I	Workflow initiator.
	M	Manually added.
	O	Last owner of task.
	R	Rule.
	U	Items to follow up.
	Y	For your information
Distr_user		User that distributed the task.
Element_type		Element type of the item.
Last_deadline		When the last deadline triggered for the task.
Lock_id		Used to detect concurrent updates of the same task.
Logged_values		Stores values chosen for logging in element type details.
Long_info		Gives information about how the task was generated.
Map_id		Uniquely identifies an instance of a step in a process.
Next_id		Points to the next task_group in the step.
	0	No next task.
Node_id		The node_id of the step.
Version_no		The version of the step.
Orig_task_id		The task_id of the predecessor task if escalated or forwarded.
Orig_user		The user that the rule found, may not be the same as wf_user_id due to fraud_detection, escalation or forwarding.
Ready_date		When the task was set to wf_status active.
Real_user		The user that performed the task.
Role_id		The role that received the task, blank if no role.
Task_group		Used together with next_id to tie together the tasks within the map_id.
Task_id		Uniquely identifies (together with map_id) a task within a step instance.

COLUMN_NAME	VALUE	DESCRIPTION
Oid		Uniquely identifies an item on workflow.
Wf_group		Groups different items on workflow (i.e. rows in an invoice).
Wf_status	ABO	Aborted.
	ACT	Active.
	AUT	Automatically approved.
	ESC	Escalated.
	DAC	Deactivated (not found in the database, shown in the workflow map when the column active is 0 for a row).
	FIN	Finished.
	FWC	Finished with changes.
	FWD	Forwarded.
	REJ	Rejected.
	TMD	Timed out.
	WTN	Waiting.
Wf_user_id		The recipient of the task, the user that is supposed to perform the task.

1.5.6 Awftaskrule

Corresponds to a rule data specification row for a task

COLUMN_NAME	DESCRIPTION
Att_parent_no	Unused.
Attribute_id	The attribute matched by the rule.
Att_value	The attribute value matched by the rule.
Field_no	The field matched by the rule.
Map_id	Uniquely identifies an instance of a step in a process.
Rel_attr_id	The attribute of the relation matched by the rule.
Rel_value	The relation value matched by the rule.
Rule_id	The rule that was matched.
Rule_type	The rule type that was matched.
Sequence_no	Used to identify a taskrule row within a task.
Task_id	Uniquely identifies (together with map_id) a task within a step instance.
Value_1	The numeric value that was matched by the rule (if any).

1.5.7 Awfmapconn

Ties together maps for connected rows

COLUMN_NAME	DESCRIPTION

Map_head	One of the maps being connected.
Map_head2	The other map being connected.

1.5.8 Awfmsvalues

Contains changes that are on master file approval

COLUMN_NAME	VALUE	DESCRIPTION
Attribute_id		The attribute id of the master file item
Change_type	1	Insert
	2	Update
	3	Delete
Master_oid		**The oid of the master file item that has changes**
Oid		Uniquely identifies a set of changes
Table_name		The main table of the master file item
Wf_group		Groups sets of changes that were performed in the same operation. If you change phone number and address on a supplier, and the approver then forwards these to two different users then the xml_data will be split and each change will get its own oid, but they will both share the same wf_group and the actual master file table will not be updated until the workflow is finished for all rows in the wf_group
Wf_state		The workflow state for the changes
Xml_data		Contains an xml with the changes

1.5.9 *fin

When workflow is finished for all rows in a wf_group the workflow is moved to these tables.

1.5.10 *histr

Not yet in use, same concept as the *fin tables

1.5.11 *wtn

Used by initial delay nodes in processes to keep track of which rows have already been processed. These tables are named by appending wtn to the table name (i.e. acutrans -> acutranswtn). They consist of the unique index in the main table (client, voucher_no, sequence_no for acutrans), as well as node_id.

1.6 Workflow service

1.6.1 Awfalertqueue

Contains queued alerts. Alerts that are sent according to a schedule will be inserted here when the alert would normally be sent (if there was no schedule), and then the workflow service will poll this table when the scheduled time is reached. Element types that are grouped (e.g. invoices and requisitions) will also get their immediate alerts queued here for 2 minutes until the start_time for the connected awftrans row with wf_action TASKALERT is reached. The reason for this is to aggregate the alerts an invoice or requisition with more than one row is approved and then all rows are sent to a new approver. Before this mechanism was introduced the next approver would get one e-mail for each row in the invoice.

If a task connected to a row in awfalertqueue is completed before the workflow service gets around to checking awfalertqueue then no alert will be sent.

COLUMN_NAME	VALUE	DESCRIPTION
Action_code		The action code for deadline alerts
Alert_instance		Unique key for awfalertqueue
Element_type		Element type of the item.
Incident		Type of alert
	NNT	Notification of new task
	NTS	Notification to substitute of new task
	SDA	Forward or escalation
	TIO	Timeout, could be reminder or escalation
Lock_id		Used to detect concurrent updates of the same alert
Map_id		Indicates which task the alert is being sent for
Scheduled	1	Scheduled alert
Task_id		Indicates which task the alert is being sent for
Wf_user_id		The recipient of the alert

1.6.2 Awftrans

The work table for the workflow service.

COLUMN_NAME	VALUE	DESCRIPTION
Trans_guid		Uniquely identifies a work item in awftrans.
Trans_head		Used to group multiple work items.

COLUMN_NAME	VALUE	DESCRIPTION
Status	N	Normal.
	E	Error, by default the workflow service will try to process this row every 15 minutes (controlled by WF_READ_E_TRANS_INTERV – see release notes for more).
Wf_action		The action the workflow service is to perform.
	ACTIONFENTRY	An action has been done from an entry step when changes were also done
	ADDSTEP	A step has been manually added to a distribution.
	ALERT	Something has been distributed online; the service sends the alert.
	BATCHDISTR	Distributes items submitted by a server job. The data to be distributed is held in a table named awfh% that was generated when the items were submitted. The data in the awfh table is divided into chunks so there can be multiple BATCHDISTR rows for a single awfh table. The default size is 100 rows in each chunk
	BATCHDROPT	Tells the workflow service to drop temporary BATCHDISTR tables (named awfh%) when there are no more BATCHDISTR rows in awftrans for the table. These rows have a start_time set to the future, so they will not be processed immediately even if the status is N
	CHANGETRD	Changes have been done on a data row that trigger a full redistribution.
	DEADLINE	A deadline has been exceeded.
	DELAY	A delay point has been reached; this work item will not be executed until start_time is reached.
	DISTRIBUTE	Something is to be distributed.
	FLUSHCACHE	Flushes the caches.

COLUMN_NAME	VALUE	DESCRIPTION
	FWC	Changes have been done on a data row that may trigger a redistribution according to the properties of the step.
	FWCCHECKOTHER	FWC has been done on another row in the same wf_group
	FWCOTHER	Changes have been done on a row in another process.
	GENENQ	Regenerates contents in the awfenquiry and awfuserlog tables for the oid
	HANDLEDEL	Rows have been deleted when distributing an existing item from an entry step
	INITSUBP	A subprocess has been reached.
	LOG	Logs a message in the workflow log (used from scripts and server jobs)
	MANDISTR	A save in manual distribution (currently only from VP10).
	MAPFIN	A map (step instance) is finished.
	MESSAGE	Used for communication between service instances in a cluster.
	MOVETOACT	Copies awfobjectmapfin rows to awfobjectmap
	MOVETOFIN	Moves all workflow items to the *fin tables for a wf_group.
	REDNEWVER	Redistributes a wf_group in a new version of the workflow process
	REDINSTEP	Redistributes a row within the current step
	SETACTION	An action has been set on a task.
	SETFINTST	A distribution has been finished; wf_state will be updated if the wf_group is finished.
	SPLIT	Something has been split.
	SYNC	A synchronisation step has been reached.
	SYSTPFIN	A system step is finished.
	SYSTPSTART	A system step will be started.
	TASKALERT	Sends an alert queued in awfalertqueue
	TASKUPDATE	Makes the workflow service order a server job to update task list values (col1-5) for a given element type
	UPDONETASKV	Updates the task list values (col1-5) for a single task

1.6.3 Awftransdet

Details for the work items in awftrans.

1.6.4 Awfservice

Used by the workflow service for load balancing, contains one row for each running workflow instance.

COLUMN_NAME	VALUE	DESCRIPTION
Bflag	1	Service instance processes deadlines.
	2	Service instance processes delays.
End_time		When other workflow services in the cluster will consider this service dead unless the timestamp is updated.
Last_update		Last time this workflow instance updated its awfservice row.
Lock_id		Used to detect concurrent updates.
Process_id		The process id of the workflow service.
Range_from		The start of the range of wf_groups processed by this service.
Range_to		Then end of the range of wf_groups processed by this service.
Server_name		The machine name where the service is running.
Service_id		A string guid uniquely identifying the service instance.
Start_time		When the service instance was started.
Status	N	Active.
	T	Terminated normally.
	E	The service instance end_time has expired, another service has set status E and the workload will be redistributed.

1.7 Enquiries

To avoid excessive joining, the data used by Enquiries, Items to Follow-up and WF User Log is precached in a data warehouse structure. The data is collected from awfobjectmap, awfmaphead, awfma, awftask and awfprocess and distilled in tables awfenquiry/fin and awfuserlog/fin. The data in these tables is updated by the Workflow Service, after certain state changing actions. Because of this, it will take a few seconds between the user, for example, approves a task and the information being shown in Enquiries becoming updated.

1.7.1 Awfenquiry

COLUMN_NAME	**DESCRIPTION**
Distr_type	Fetched from *awftask*
Element_type	Fetched from *awfmaphead*
Error_no	Fetched from *awfobjectmap*
Log_id	Unique identifier of an entry
Map_head	Fetched from *awfmaphead*
Map_id	Fetched from *awfmap*
Oid	Fetched from *awfobjectmap*
P_description	Fetched from *awfprocess* The description of the process
Proc_node_id	Fetched from *awfprocess* Node_id of the process
S_description	Fetched from *awfprocess* The description of the current step
Step_node_id	Fetched from *awfprocess* Node_id of the current step
Task_id	Fetched from *awftask*
Version_no	Fetched from *awfprocess* Current version of both the process and the step
Wf_group	Fetched from *awfobjectmap*
Wf_user_id	Fetched from *awftask*

1.7.2 Awfuserlog

COLUMN_NAME	**DESCRIPTION**
Action_code	Fetched from *awftask*
Distr_type	Fetched from *awftask*
Element_type	Fetched from *awfmaphead*
Log_id	Unique identifier of an entry
Map_head	Fetched from *awfmaphead*
Map_id	Fetched from *awfmap*
Oid	Fetched from *awfobjectmap*
P_description	Fetched from *awfprocess* The description of the process

COLUMN_NAME	DESCRIPTION
Proc_node_id	Fetched from *awfprocess* Node_id of the process
Real_user	Fetched from *awftask*
S_description	Fetched from *awfprocess* The description of the current step
Step_node_id	Fetched from *awfprocess* The description of the current step
Task_id	Fetched from *awftask*
Version_no	Fetched from *awfprocess* Current version of both the process and the step
Wf_group	Fetched from *awfobjectmap*
Wf_user_id	Fetched from *awftask*

1.7.3 *fin

When workflow is finished all data is removed from the active tables and re-generated in the fin tables.

1.7.4 Regeneration

The Workflow Service can be instructed to regenerate all or parts of the data in the precache. It can be achieved by inserting instructions into awftrans with one of the following queries. Note that you also have to set both bflag to 8 and specify an argument:

To regenerate only active enquiry and userlog rows:

```
INSERT INTO awftrans (argument, bflag, priority, trans_guid,
wf_action, status, wf_group) VALUES ('REGENERATE_ACTIVE', 8, -
151, GETGUID(), 'GENENQ', 'N', '')
```

To regenerate only finished enquiries and userlog rows:

```
INSERT INTO awftrans (argument, bflag, priority, trans_guid,
wf_action, status, wf_group) VALUES ('REGENERATE_FINISHED', 8,
-152, GETGUID(), 'GENENQ', 'N', '')
```

To regenerate **all** enquiry and userlog rows:

```
INSERT INTO awftrans (argument, bflag, priority, trans_guid,
wf_action, status, wf_group) VALUES ('REGENERATE_ALL', 8, -
150, GETGUID(), 'GENENQ', 'N', '')
```

An individual item can be regenerated by providing the item's oid, but with bflag set to 0 and argument set to " (empty).

1.8 Items that are on workflow

1.8.1 *map

All items that are on workflow are assigned a guid that is stored in these map tables. The map tables are named by appending map to the table name (i.e. acrtrans -> acrtransmap). The map tables consist of the unique index in the main table (client, voucher_no, sequence_no for acrtrans), as well as the oid column with an unique index on both.

1.8.2 Workflow item tables

The table in an element type that has blank table_det column in asys/wfelemtype will normally have a wf_state column. Unless "Leave workflow status unchanged" has been selected for å process this column will be updated to W while an item is on workflow. If it is rejected while it is on workflow it will be set to R. When all the awfmaphead rows with the same wf_group are done the wf_state will be updated to N,T or C according to what happened in the workflow.

COLUMN_NAME	VALUE	DESCRIPTION
Wf_state	(blank)	No workflow
	A	Row upgraded from 5.4
	C	Row is rejected, workflow is finished.
	N	No workflow.
	T	Row is approved, workflow is finished
	R	Row is rejected, but workflow is not finished yet. This will change to W once the row is approved by someone in the workflow
	U	Row is in "Items to follow up"
	W	Row is on workflow
	X	Row is not yet sent to workflow

2 Known Issues

2.1 Caching

Element type setup, and most other fixed data is cached. If changes are made to these values then either restart the servers, or make sure that the caches are flushed.

To flush the cache in the workflow service, run this SQL:

MSSQL:

Insert into awftrans (wf_action,status,trans_guid,wf_group) values ('FLUSHCACHE','N',newid(),'X').

Oracle:

Insert into awftrans (wf_action,status,trans_guid,wf_group) values ('FLUSHCACHE','N', sys_guid(),'X').

2.2 Element type details

If changes have been made to what data is displayed in the task list then the “Update task list values” tool menu function should be used (exists in 5.5.3 update 6 and newer versions). In earlier versions it is not advisable to make these changes.

2.3 Delay nodes

New delay nodes in a process will not be used right away. Delay nodes are processed by a background thread in the workflow service, and by default this thread will check every 30 minutes if there are new delay nodes to check (controlled by WF_DELAY_RELOAD_INTERVAL). For delay nodes that are within the workflow process (i.e. not at the beginning) the background thread will only check versions that have active rows in a given delay, so these can take up to 30 minutes to be processed even if the delay node is old.

A given row in a table will only be picked up by an initial delay node once for a given workflow process. This is controlled by the *wtn tables. This means that having an initial

delay node to trigger a process that can happen more than once is a bad idea (i.e. it will not work).

2.4 Deadlines

The workflow service has a background thread that checks deadlines. It is not always easy to predict how often a deadline will be checked. There are several factors that play a role:

1. The workflow service will check every 30 minutes if there are active steps with deadlines that need to be checked. The reason for this is that different versions of the process can have different deadlines, and since old versions of the process can have active workflow for a long time we have to continue checking deadlines on the old versions as well. By doing it this way we minimize the work the workflow service needs to do, while still honoring all deadlines. To make the workflow service check more or less often for active steps you can create and set the common parameter WF_TIMER_RELOAD_INTERVAL to a value (in minutes) and restart the workflow service.
2. The workflow service will look at how long the deadline is and divide this by the value in the system parameter WF_DEADLN_FACTOR (default value 24). If for example the deadline is 12 hours, the deadline will be checked every 12 (hours) / 24 (deadline factor) = 0.5 hours = 30 minutes. This is how often the deadline will be checked, with two exceptions.
    a. The deadline will not be checked more often that the value of the system parameter WF_DEADLN_LOWER_THRESHOLD (time in minutes, decimals allowed). The default value is 5 minutes, so no deadline will be checked more often than every 5 minutes unless this parameter is changed.
    b. If the value is larger than the value of the system parameter WF_DEADLN_UPPER_THRESHOLD (time in minutes, decimals allowed), then the deadline will only be checked once every night. The default value is 60 minutes, so any deadline of more than 60 minutes * 24 (deadline factor) = 1440 minutes = 1 day will only be checked once every night, if the system is running with default values for all parameters.
        i. The system parameter WF_DEADLN_NIGHTLY_CHECK determines when the nightly check is run. The default is “01:45”, and the time needs to be written thus: “hours:minutes”.

2.5 Counters

If invoice manager upgrade has been run incorrectly or if workflow setup has been copied between clients then you may run into a situation where the counters that control rule_id (WF_RULE) and node_id (WFPROCESS_NEXTNODE) get out of sync with what is in the database. If new nodes in awfprocess have node_ids less than 10.000 you probably have a problem.

2.6 Process designer is blank or missing some processes

We have had cases where the process designer tree is completely blank. This is caused by no rows in awfprocess with parent_id -1 having first_flag set. We think this happens if you move processes or branches in the process tree. This has been patched, but there have been reports after that as well. We are not sure if this is caused by unpatched clients or if there still are bugs in the code that causes this. This can also be caused by upgrade errors.

The same thing can happen for a process in the process designer canvas, i.e. no rows have first_flag set, so no process is drawn.

There have also been cases where some processes are missing from the process designer. That happens if no rows have their next_id pointing to the node_id of the missing row.

How to fix blank process designer or missing rows:
If nothing shows at the top level of the process designer tree then you need to look at all the rows with parent_id -1, and figure out which one you want to be first - set first_flag to 1 on that one, and 0 on all others. Set the next_id of the first row to point to the node_id of the next row (and the previous_id for the next row to point back), and do the same for all subsequent rows. Set next_id to -1 on the last row.

If you have missing processes then you need to go through the points above and modify the next_id/previous_id of some rows so that the missing process is included in the tree.

If you have more than a handful of rows to organize it can be helpful to write the node_id of each row on a post-it, lay them out one after another and use that as a guide when you update and verify next/previous_id.

If you are missing nodes on a deeper level of the process designer tree then you need to figure out the node_id X of the branch, and then use the method above for all rows with parent_id X. You can also use the method above to fix processes, as long as you keep in mind that you also have to filter by version_no.

2.7 The layout in the process designer is all wrong

We have had cases where the layout in the process designer looks weird. This happens if the process designer is unable to load the layout stored in aimblob.
The layout is stored in the table aimblob, and the way that the process designer finds the layout is by using the blob_id in the table acrdiagramlayout by looking up the node_id and version of the process in the diagram_id column (it is on this format: "node_id;version_no"). The easiest way to find the node_id of the process is to look in awfprocelemtype using the element type.

Try checking if anything is missing from acrdiagramlayout or aimblob. There have been cases where rows were missing from acrdiagramlayout, and the layout was restored when we fixed this. In this case AG09 had previously been run to copy processes, and something strange had happened because there were lots of rows in acrdiagramlayout referencing the missing layout, except on the client is was supposed to be on.

If everything looks ok in acrdiagramlayout and aimblob, then try just creating a new version of the process. That worked on another process at the same site with the missing row in acrdiagramlayout.

2.8 Tasks are displayed in the task list, but are missing in the approval screen

What is displayed in the tasklist is only based on the contents of awftask, while what is displayed in the approval screen uses data from awftask, the *map table (e.g. acrtransmap), and the transaction tables.
There are several things that can cause missing tasks in the approval screen:

1) The item on workflow has been deleted. Start in awftask and find the oid there. Look up this oid in the mapping table (e.g. acrtransmap). If you find a row in the mapping table, use the unique key from there to see if you find the item in the transaction table(s) (e.g. acrtrans). If you can’t find the row in the mapping table, then you can try

to look up the oid in awfobjectmap and use the composite_key and map_table_name values when trying to figure out what has happened.

2) The values that are displayed in the task list do not match the values in the transaction tables. Values in col1-5_value from awftask are displayed in the task list. When a task list item with extra values is selected, these values are used to narrow down the list of items that are displayed in the approval screen. Example: Values from dim_1 which contain costcenters are displayed in the task list. The user clicks on the tasks for costcenter 210, but nothing is displayed because acrtrans.dim_1 has been updated without the corresponding workflow code being called. To fix this issue you go into the Element types screen, select the corresponding element type, go into Element type details and click tool item “Update task list values”. The col1-5_value columns in awftask will then be updated by the workflow service.

3 Tips and tricks

3.1 Logging

In Agresso 5.6 the standard configuration in the management console controls logging.
The information below on logging is only applicable to 5.5.3 and earlier:
To get query logging in the workflow log you can either set the common parameter WF_LOGLEVEL to "console=4", or set the business server property WF_LOGLEVEL to "console=4". After you have set this parameter you must restart the workflow service.

The different values possible are:

- "console=1" – Errors
- "console=2" – Warnings
- "console=3" – Information
- "console=4" – Queries

There is a bug in SP1 that makes it necessary to set some additional parameters to get query logging. Create these common parameters with the values specified below and restart the workflow service, but make sure you turn them off after you are done recreating the problem because they force the workflow service to not use worker threads:

Name: WF_MAX_THREADS, **value:** 1
Name: WF_SERVICE_ASYNC, **value:** -1

3.2 Aggregation

To aggregate all the values in an item (e.g. all the rows in an invoice), set WF26 to aggregate on CLIENT (A3).

3.3 Debugging

To debug what the workflow service does step by step you can start the workflow service from the command line:
WorkflowService -sDEBUGU serverdatasourcename
(processes rows with status U)

5.6+:

WorkflowService –sDEBUGUX serverdatasourcename

(processes rows with status U, any new awftrans rows will have status X)

WorkflowService –sDEBUGUU serverdatasourcename

(processes row with status U, any new awftrans rows will have status U so there is no need to do any manual updates to continue processing rows for a given item).

This will make the workflowservice only process rows in awftrans that have status U (you can choose any letter you like, except N or E). New awftrans rows get status N (or any other status you specify in 5.6+) so these will not be processed until you manually update their status to U. This will enable you to check in the workflow tables when a given status is updated, rows are inserted or deleted or anything else that you need to figure out. Stop the workflow service by typing EXIT (in all caps) and hitting return. When you run the workflow service in this way you will not get anything in the workflow log. It will not check deadlines or delay nodes, and you will not find a row in awfservice for the running instance. This means that if there is a normal workflow service running against the database it will not be disturbed, but you need to be quick when you update the status from N to something else for new rows or the normal workflow service will pick them up.

To debug deadlines use this switch: **-sDEADLINE**.
This will run through one pass of checking deadlines and then stop.

To debug delay nodes use this switch: **-sDELAY**.
This will run through one pass of checking delay nodes and then stop.

To debug alerts use this switch: **-sALERT**
This will run through one pass of checking if there are any alerts in awfalertqueue that should be sent and then stop.

To debug cluster validation use this switch: **-sCLUSTERVALIDATION**
This will force the cluster validation code to be run

3.4 Regular expressions

Starting in 5.6 it is possible to use regular expressions in OR-splits and data collection in rules. Agresso uses the .Net variant of regular expressions. Regular expressions are very powerful and this makes it possible to do lots of checks in rules and OR-splits that previously would have required writing ACT code.

Below is a subset of regular expression syntax, there are many more operators that can be used:
^ - matches the start of a string
$ - matches the end of a string
. – matches any character
* - will match zero or more instances of the preceding character (or group of characters)
+ - will match one or more instances of the preceding character (or group of characters)

Here are some simple examples:
If you want a rule to match any string containing BC anywhere then just use BC (it will match BC, OJIOJBCJOJO, JBC, BCEEE, and so on):
BC

If want to only match strings with three characters you use the following expression (will match 123, ABC, JOJ, EO1, 3HC, and so on):
^…$

If you want to match any string starting with B use the following (will match B, BE, B1, BOJIEOJ, and so on):
^B

If you want to match any string starting with B and ending with C use the following (it will match BC, BAC, BOJIEOJJC, and so on):
^B.*C$

For more advanced examples just google regular expressions or buy a book on them (Amazon has a good selection).

3.5 Threading

The workflow service is multithreaded in order to better utilize multiple cores/CPUs. Here is a brief overview of the different threads that exist:

- The main thread: reads new entries from awftrans. If WF_SERVICE_ASYNC is set to 0 or -1 at the same time as WF_MAX_THREADS is 1, this thread will also do the actual work

- Worker threads: By default there are 4, but this can be overridden by WF_MAX_THREADS. If WF_MAX_THREADS is 1 and WF_SERVICE_ASYNC is set to 0 or -1 then there will be no worker threads, everything will happen in the main thread

- Deadline thread: This thread reads the deadline setup and periodically checks deadlines

- Delay thread: This thread reads the delay setup and periodically checks delay nodes.

- Alert thread: This thread sends alerts that have been set up to use a schedule.

For maximum performance we recommend setting WF_MAX_THREADS equal to the total number of CPU cores in the server. E.g. if you have 2 quad core CPUs WF_MAX_THREADS should be set to 8 to process as much as possible. WF_MAX_THREADS can be set higher than the number of cores, but the total throughput of the system will be more or less the same as if it had been set equal. Please note that most installations will not need to tune this parameter, as the workflow service seldom is the performance bottleneck.

3.6 Multiple workflow services in the same database

It is easy to have multiple workflow services in the same database, just set up an additional workflow queue on a separate server and start it. The new workflow service and the existing one will detect each other's presence using awfservice and automatically redistribute the workload so that each service will process a separate range of wf_group values from awftrans. If one of the services stops the other will detect this after a little while and take over the full workload.

4 Appendix

4.1 ACT named events (5.5.3+)

4.1.1 OR splits

By making a named event in the event family WF_RULE it is possible to implement OR split functions in ACT code. The event name must be equal to the method_name chosen in awfblmethods. Assembly_name and class_name must be left blank for ACT functions. The other columns in awfblmethods must be filled out as for normal functions:

Column name	Datatype	Description
assembly_name	string	The assembly the function is found in for non-ACT functions
class_name	string	The class the function is found in for non-ACT functions
data_type	string	The datatype returned by the function, possible values: b – bool s – string d – double i - int
element_type	string	The element type that can use the function
method_id	int	Unique key, must be unique across asyswfblmethods and awfblmethods. It is suggested that user defined functions start at 10000 or above.
method_name	string	The name of the function
title_no	int	The title that is displayed in the process designer when choosing functions

These parameters will be passed to the event:

Parameter name	Datatype	Description
row	DataRow	the row being processed
data	DataSet	the dataset containing the row being processed
client	string	the client of the row being processed
element_type	string	the element_type of the row being processed
node_id	Int	the node_id of the OR-split
version_no	int	the version_no of the OR-split
method_name	String	the name of the called method

The result of the method must be passed in a parameter named result, and must be of the same type as specified in awfblmethods. If the event is canceled the rules engine will handle this as if the function did not match the given value.
Example OR split ACT event:

```
    /// <summary>
   /// Unit test for named events
```

```
/// </summary>
[NamedEvent("WF_RULE", "GetDoubleTestValue", "Test.WorkFlow unit test")]
public class GetDoubleTestValueListener : IProjectNamedEvent
{
    INamedHooks _event;

    #region IProjectNamedEvent Members

    public void Initialize(INamedHooks events)
    {
        _event = events;
        _event.OnNamedEvent += new EventHandler<NamedEventArgs>(EventRaised);
    }

    void EventRaised(object sender, NamedEventArgs e)
    {
        e.ReturnValues.Cancel = false;
        DataRow row = e.Parameters["row"] as DataRow;
        Assert.IsNotNull(row, "Verify that the row is passed");
        DataSet ds = e.Parameters["data"] as DataSet;
        Assert.IsNotNull(ds, "Verify that the dataset is passed");
        e.ReturnValues["result"] = 43.2;
    }

    #endregion
}
```

4.1.2 System steps

By making named events in the event family WF_SERVICE it is possible to implement system steps in ACT code. The base event name must be equal to what is entered in menu_id_web in awfelemtypemenu.
The other columns in awfelemtypemenu must be filled out as for a normal system step:

Column name	Datatype	Description
element_type	string	The element type that can use the system step
entry_flag	int	Indicates if this is an entry screen, must be 0 for system steps
menu_id_office	string	The menu_id to be used when an approval screen is opened in the smart client. For normal system steps contains the menu_id of the report to run. Must be blank for ACT system steps.
menu_if_web	string	The menu_id to be used when an approval screen is opened in self service. Enter the base event name here for ACT system steps.
menu_ref	int	Unique key, must be unique across asyselemtypemenu and awfelemtypemenu. It is suggested that user defined items start at 10000 or above.
s_usage	string	The type of menu item, valid values: WFS – screen (user step) WFR – Report (system step), use this for ACT system steps
title_no	int	The title that is displayed in the process designer when choosing functions

When a system step ACT event is triggered there are actually three events that will be triggered. If you choose UpdateOther as your base event name, then the event before_UpdateOther will be triggered first, before the transaction begins. If your event needs to read data and do processing you should do that here. When before_UpdateOther is finished the workflow service will begin a transaction and then trigger the UpdateOther event. After

the UpdateOther event has finished workflow will do inserts and updates that will make workflow continue and then commit the transaction. When the transaction has commited the event after_UpdateOther will be triggered.

All the events have the same set of parameters:

Parameter name	Datatype	Description
table_name	string	the table in the dataset containing the rows to process
data	DataSet	the dataset with the rows to process
client	string	the client of the rows to processed
element_type	string	the element_type of the rows to process
node_id	Int	the node_id of the system step
version_no	int	the version_no of the system step

There are no return values from the events, and they cannot be canceled. If you need to read extra data for use within the transaction this data can be put into the DataSet passed to the event. You only have to override the events you actually need. When the event is triggered the DataSet passed can contain many unrelated rows, so the ACT code must not assume that there is only one row in the DataSet.

Example ACT systemp step:

```
[NamedEvent("WF_SERVICE", "before_TestSystemEvent", "Test.Workflow unit test")]
public class BeforeTestSystemEventListener : IProjectNamedEvent
{
    INamedHooks _event;

    #region IProjectNamedEvent Members

    public void Initialize(INamedHooks events)
    {
        _event = events;
        _event.OnNamedEvent += new EventHandler<NamedEventArgs>(EventRaised);
    }

    void EventRaised(object sender, NamedEventArgs e)
    {
        DataSet ds = e.Parameters["data"] as DataSet;
        string tableName = e.Parameters["table_name"] as string;
        foreach (DataRow row in ds.Tables[tableName].Rows)
        {
            row["address"] = "somestring";
        }
    }

    #endregion
}

/// <summary>
/// Unit test for named events
/// </summary>
[NamedEvent("WF_SERVICE", "TestSystemEvent", "Test.Workflow unit test")]
public class TestSystemEventListener : IProjectNamedEvent
{
    INamedHooks _event;

    #region IProjectNamedEvent Members

    public void Initialize(INamedHooks events)
    {
        _event = events;
        _event.OnNamedEvent += new EventHandler<NamedEventArgs>(EventRaised);
    }

    void EventRaised(object sender, NamedEventArgs e)
```

```
    {
        DataSet ds = e.Parameters["data"] as DataSet;
        string tableName = e.Parameters["table_name"] as string;
        foreach (DataRow row in ds.Tables[tableName].Rows)
        {
            // do updates
        }
    }

    #endregion
}

/// <summary>
/// Unit test for named events
/// </summary>
[NamedEvent("WF_SERVICE", "after_TestSystemEvent", "Test.Workflow unit test")]
public class AfterTestSystemEventListener : IProjectNamedEvent
{
    INamedHooks _event;

    #region IProjectNamedEvent Members

    public void Initialize(INamedHooks events)
    {
        _event = events;
        _event.OnNamedEvent += new EventHandler<NamedEventArgs>(EventRaised);
    }

    void EventRaised(object sender, NamedEventArgs e)
    {
        DataSet ds = e.Parameters["data"] as DataSet;
        string tableName = e.Parameters["table_name"] as string;
        foreach (DataRow row in ds.Tables[tableName].Rows)
        {
            // do things you want done after the transaction
        }
    }

    #endregion
}
```

4.2 New Workflow ACT named events (5.6 SP1+)

Starting in 5.6 SP1 workflow supports a number of new ACT named events. They all correspond to standard events. The behavior in the standard events can be cancelled by setting the ReturnValues.Cancel property of NamedEventArgs. The event family of the events is WF_CONTROL. The event name is the element type + underscore (“_”) + the name of the method. If you for example want to override the BeforeWFFinished method for supplier invoices, the event name will be SIN_BeforeWFFinished. Before workflow calls the methods below there will be one dummy call to check if there is any listeners on the ACT event. This is because some of these events requires the workflow service to read extra data, and this is skipped for performance reasons when there are no listeners. These dummy calls can be identified by checking if the parameter is_in_use_check has been set. If this parameter has been set then ignore this event.

4.2.1 BeforeWFFinished and WFFinished

Parameter name	Datatype	Description
client	String	The client of the transaction
dc	DataSet	The item(s) that finish workflow
saveData (only BeforeWFFinished)	DataSet	Contains the data that was submitted to workflow in cases where BeforeWFFinished is triggered immediately (i.e. when there is no workflow)

These events occur when workflow is finished and wf_state is updated on the tables that are on workflow. BeforeWFFinished is triggered before the database transaction starts, so put any SELECTs inside this event. The results of the SELECTs can be stored in the dc DataSet and then be read from the DataSet inside the WFFinished event, which occurs inside the transaction. The dc DataSet will only contain data from the table with the wf_state column in the element type. The wf_state indicates the result of the workflow. If the transaction is grouped then the table will contain all the grouped rows.

Wf state	Description
N	No workflow
T	Approved
C	Rejected / Aborted

Example implementations of BeforeWFFinished and WFFinished:

```
[NamedEvent("WF_CONTROL", "SIN_BEFOREWFFINISHED", "Test.WorkFlow unit test")]
   public class TestSINBeforeWFFinishedListener : IProjectNamedEvent
   {
       INamedHooks _event;

       #region IProjectNamedEvent Members

       public void Initialize(INamedHooks events)
       {
           _event = events;
           _event.OnNamedEvent += new EventHandler<NamedEventArgs>(EventRaised);
       }

       void EventRaised(object sender, NamedEventArgs e)
       {
           const string is_in_use_check = "is_in_use_check";
           object isInUse = e.Parameters[is_in_use_check];
           if (isInUse != null)
           {
               return;
           }

           const string acrtrans = "acrtrans";
           const string wf_state = "wf_state";
           const string rejected = "C";
           const string amount = "amount";
```

```
            DataSet ds = e.Parameters[WFMethodParams.DcParam] as DataSet;
            foreach (DataRow row in ds.Tables[acrtrans].Rows)
            {
                if (row[wf_state].ToString() == rejected)
                {
                    // do some heavy calculations, read something from the database or a web service  in
memory and put the result into DC
                }
            }

            e.ReturnValues[WFMethodParams.ReturnParam] = true;
        }

        #endregion
    }

    [NamedEvent("WF_CONTROL", "SIN_WFFINISHED", "Test.WorkFlow unit test")]
    public class TestSINWFFinishedListener : IProjectNamedEvent
    {
        INamedHooks _event;

        #region IProjectNamedEvent Members

        public void Initialize(INamedHooks events)
        {
            _event = events;
            _event.OnNamedEvent += new EventHandler<NamedEventArgs>(EventRaised);
        }

        void EventRaised(object sender, NamedEventArgs e)
        {
            const string is_in_use_check = "is_in_use_check";
            object isInUse = e.Parameters[is_in_use_check];
            if (isInUse != null)
            {
                return;
            }
            const string acrtrans = "acrtrans";
            const string wf_state = "wf_state";
            const string rejected = "C";
            const string client = "client";
            const string voucher_no = "voucher_no";
            const string sequence_no = "sequence_no";

            DataSet ds = e.Parameters[WFMethodParams.DcParam] as DataSet;

            foreach (DataRow row in ds.Tables[acrtrans].Rows)
            {
                if (row[wf_state].ToString() == rejected)
                {
                    // do some kind of update
                    Sql update = "UPDATE acrtrans SET amount=0 WHERE client=@client AND
voucher_no=@voucher_no AND sequence_no=@sequence_no";
                    update[client] = row[client];
                    update[voucher_no] = row[voucher_no];
                    update[sequence_no] = row[sequence_no];
                    DatabaseContext.Current.Execute(update);
```

```
            }
        }
    }

    #endregion
}
```

4.2.2 BeforeWFStarted and WFStarted

Parameter name	Datatype	Description
client	String	The client of the transaction
dc	DataSet	The item(s) that start workflow
saveData (only BeforeWFStarted)	DataSet	Contains all the data that was submitted to workflow in cases where BeforeWFStarted is triggered immediately (i.e. when there is no workflow)

These events occur when workflow starts. BeforeWFStarted is triggered before the database transaction starts, so put any SELECTs inside this event. The results of the SELECTs can be stored in the dc DataSet and then be read from the DataSet inside the WFStarted event, which occurs inside the transaction. The dc DataSet will only contain data from the table with the wf_state column in the element type. If the transaction is grouped then the table will contain all the grouped rows.

4.2.3 BeforeWFSaved and WFSaved

Parameter name	Datatype	Description
client	String	The client of the transaction
wfData	DataSet	The item(s) that were submitted to workflow
saveData (only BeforeWFSaved)	DataSet	Contains all the data that was submitted to workflow

These events occur when items are submitted to workflow. BeforeWFSaved is triggered before the database transaction starts, so put any SELECTs inside this event. The results of the SELECTs can be stored in the wfData DataSet and then be read from the DataSet inside the WFSaved event, which occurs inside the transaction. The wfData DataSet will only contain data from the table with the wf_state column in the element type. If the transaction is grouped then the table will contain all the grouped rows.

4.2.4 BeforeUpdate and Update

Parameter name	Datatype	Description
container	IMasterFileUpdateContainer	The changes that are going to be updated. Please see documentation of the IMasterFileUpdateContainer interface for details
dc	DataSet	A DataSet that can be used to store extra data between the BeforeUpdate and Update calls

These events occur when values are updated in element types that have master file approval. For fields that do not have workflow, and in cases where no workflow has been configured these events occur in the same transaction as the normal save. For fields that have workflow these events only occur when all fields in an update have finished workflow. BeforeUpdate is triggered before the database transaction starts, while Update is triggered inside the transaction (but before the changes in container have been performed). The IMasterFileUpdateContainer interface is defined in Agresso.Interface.Fundamentals.Workflow.dll.

4.2.4.1 IMasterFileUpdateContainer methods

IList<string> GetUpdatedTables()
Lists the tables that have updates

IList<IMasterFileValuesForUpdate> GetUpdates(string table)
Lists all the updates for a table

Argument name	Explanation
table	The table to retrieve updates for

string GetMainMasterFileTable();
Retrieves the main table in the masterfile

4.2.4.2 ImasterFileValuesForUpdate methods

ChangeTypeEnum GetChangeType()
Retrieves the ChangeType for the update

void SetChangeType(ChangeTypeEnum changeType)
Sets a new change type on the update, can be used to cancel the update by setting ChangeTypeEnum.None

Argument name	Explanation
changeType	The new change type

IList<string> GetChangedFields()
Lists all the fields that have been updated

object GetChangedValue(string field)
Retrieves the new value for a changed field

Argument name	Explanation
field	The field to retrieve the value for

IList<string> GetVirtualFields()
Lists all the virtual fields whose name starts with MasterFileApprovalEntity.PersistVirtualColumnInMfaPrefix that had a value when the change was sent on workflow.

string GetVirtualValue(string field)
Retrieves the value for a virtual field whose name starts with MasterFileApprovalEntity.PersistVirtualColumnInMfaPrefix that had a value when the change was sent on workflow.

Argument name	Explanation
field	The virtual field to retrieve the value for

object GetCurrentKeyValue(string field)
Retrieves the current value for a key field

Argument name	Explanation
field	The field to retrieve the value for

IRow GetRow()
Retrieves an IRow with all changed values the way it will look when it is updated

IEntityDefinition GetEntityDefinition()
Retrieves the entity definition for the row if it has been defined in asyselemtypedet, null otherwise

string GetAttributeId()
Retrieves the attribute id for the master file

4.2.5 UpdateOtherChangesIfNeeded

Parameter name	Datatype	Description
currentChange	IMasterFileUpdateContainer	The changes that are going to be updated. Please see documentation of the IMasterFileUpdateContainer interface for details
otherChange	IMasterFileUpdateContainer	Other master file approval changes that are on workflow

In cases where the changes that are performed to a master file cause rows to be deleted or otherwise changed, it may be necessary to change or remove other changes to the same master file item from workflow. If for example a changed value in one column means that rows in another table in the master file should be deleted, and the rows that are deleted also have changes that are on master file approval, then the changes on the deleted rows should be removed from workflow. This is handled by this event, by setting the changetype on the data in otherchange to None.

4.2.6 BeforeSystemStep, SystemStep and AfterSystemStep

Parameter name	Datatype	Description
systemStep	string	The id of the systemstep
client	string	The client
nodeId	int	The node_id of the systemstep
versionNo	int	The version_no of the systemstep
elementType	string	The element type of the system step
tableName	string	The table in dc that contains the data to be processed by the system step
dc	DataSet	The DataSet that contains the data to be processed by the system step

This event is triggered for system steps that are defined in .Net code (and not for regular system steps that are run in Server processes). BeforeSystemStep is triggered before the transaction begins, SystemStep is triggered inside the transaction and AfterSystemStep is triggered after the transaction has been committed.