

Progress Dynamics. Repository Reference

© 2005 Progress Software Corporation. All rights reserved.

Progress® software products are copyrighted and all rights are reserved by Progress Software Corporation. This manual is also copyrighted and all rights are reserved. This manual may not, in whole or in part, be copied, photocopied, translated, or reduced to any electronic medium or machine-readable form without prior consent, in writing, from Progress Software Corporation.

The information in this manual is subject to change without notice, and Progress Software Corporation assumes no responsibility for any errors that may appear in this document.

The references in this manual to specific platforms supported are subject to change.

A [Stylized], Allegrix, Allegrix & Design, Business Empowerment, eXcelon, ObjectStore, PeerDirect, Progress, Progress Dynamics, Powered by Progress, Empowerment Center, Progress Empowerment Center, Progress Empowerment Program, Progress Fast Track, Progress OpenEdge, Progress Profiles, Partners in Progress, Partners en Progress, Progress en Partners, Progress in Progress, P.I.P., Progress Results, Progress Software Developers Network, ProVision, ProCare, ProtoSpeed, SmartBeans, SpeedScript, Technical Empowerment, and WebSpeed are registered trademarks of Progress Software Corporation or one of its subsidiaries or affiliates in the U.S. and/or other countries. AccelEvent, A Data Center of Your Very Own, AppsAlive, AppServer, ASPen, ASP-in-a-Box, BusinessEdge, Cache-Forward, Fathom, Future Proof, IntelliStream, ObjectCache, ObjectStore Event Engine, ObjectStore RFID Accelerator, ObjectStore Trading Accelerator, OpenEdge, POSSE, POSSENET, ProDataSet, Progress Business Empowerment, Progress for Partners, PSE Pro, PS Select, SectorAlliance, SmartBrowser, SmartComponent, SmartDataBrowser, SmartDataObjects, SmartDataView, SmartDialog, SmartFolder, SmartFrame, SmartObjects, SmartPanel, SmartQuery, SmartViewer, SmartWindow, WebClient, and Who Makes Progress are trademarks or service marks of Progress Software Corporation or one of its subsidiaries or affiliates in the U.S. and other countries.

Java and all Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

Any other trademarks and service marks contained herein are the property of their respective owners.

February 2005



Product Code: 4480
Item Number: 103617;V2.1B

Contents

Preface	xv
Purpose	xv
Audience	xv
Organization of this manual	xv
How to use this manual	xvii
Typographical conventions	xviii
Syntax notation	xix
Progress messages	xxii
1. The Progress Dynamics Repository	1-1
1.1 Repository model	1-2
1.2 Repository conventions	1-4
1.2.1 Schema definition files	1-4
1.2.2 Table conventions	1-5
1.2.3 Field conventions	1-6
2. Customization Group	2-1
2.1 Customization group structure	2-2
2.2 Managers	2-2
2.3 Table descriptions	2-3
2.3.1 ryc_customization_result table—ryccr	2-3
2.3.2 ryc_customization_type table—ryccy	2-4
2.3.3 ryc_render_type table—rycrt	2-5
2.3.4 rym_customization table—rymcz	2-6

3.	Deployment and Versioning Group	3-1
3.1	Deployment and Versioning group structure	3-2
3.2	Managers	3-3
3.3	Table descriptions	3-3
3.3.1	gst_deployment table—gstdp	3-4
3.3.2	gsc_dataset_entity table—gscde	3-5
3.3.3	gsc_deploy_dataset table—gscdd	3-6
3.3.4	gsc_deploy_package table—gscdp	3-8
3.3.5	gsc_package_dataset table—gscpd	3-8
3.3.6	gsc_scm_tool table—gscsm	3-9
3.3.7	gsm_release table—gsmrl	3-10
3.3.8	gsm_scm_xref table—gsmsx	3-11
3.3.9	gst_dataset_file table—gstdf	3-13
3.3.10	gst_record_version table—gstrv	3-14
3.3.11	gst_release_version table—gstrl	3-15
3.3.12	ryt_dbupdate_status table—rytds	3-16
4.	Entity and Defaults Group	4-1
4.1	Entity and Defaults group structure	4-2
4.2	Table descriptions	4-4
4.2.1	gsc_entity_mnemonic table—gscem	4-5
4.2.2	gsc_data_tag table—gsctg	4-6
4.2.3	gsc_default_code table—gscdc	4-7
4.2.4	gsc_default_set table—gscds	4-8
4.2.5	gsc_default_set_usage table—gscdu	4-9
4.2.6	gsc_entity_display_field table—gsced	4-9
4.2.7	gsm_control_code table—gsmcl	4-11
4.2.8	gsm_entity_field table—gsmeff	4-12
4.2.9	gsm_entity_field_value table—gsmev	4-12
4.2.10	gsm_external_xref table—gsmex	4-13
4.2.11	gsm_filter_data table—gsmfd	4-15
4.2.12	gsm_filter_set table—gsmfi	4-16
4.2.13	gsm_tagged_data table—gsmtd	4-17
4.2.14	ryc_relationship table—rycre	4-18
4.2.15	ryc_relationship_field table—rycrf	4-20
4.2.16	ryc_ri_default table—rycri	4-21
5.	Error Group	5-1
5.1	Error group structure	5-2
5.2	Table descriptions	5-2
5.2.1	gsc_error table—gscer	5-2
5.2.2	gst_error_log table—gster	5-4

6.	Globalization Group	6-1
6.1	Globalization group structure	6-2
6.2	Managers	6-3
6.3	Table descriptions	6-3
6.3.1	gsc_language table—gsclg	6-4
6.3.2	gsc_global_control table—gscgc	6-5
6.3.3	gsc_global_default table—gscgd	6-6
6.3.4	gsc_language_text table—gscit	6-7
6.3.5	gsc_nationality table—gscna	6-8
6.3.6	gsm_country table—gsmcy	6-9
6.3.7	gsm_currency table—gsmcr	6-10
6.3.8	gsm_translation table—gsmtl	6-11
7.	Menu and Toolbar Group	7-1
7.1	Menu and Toolbar group structure	7-2
7.2	Managers	7-4
7.3	Table descriptions	7-4
7.3.1	gsm_menu_item table—gsmmi	7-4
7.3.2	gsc_instance_attribute table—gscia	7-6
7.3.3	gsc_item_category table—gscic	7-8
7.3.4	gsm_menu_structure table—gsmms	7-8
7.3.5	gsm_menu_structure_item table—gsmi	7-10
7.3.6	gsm_object_menu_structure table—gsmom	7-10
7.3.7	gsm_toolbar_menu_structure table—gsmtm	7-12
7.3.8	gsm_translated_menu_item table—gsmti	7-12
8.	Module Group	8-1
8.1	Module group structure	8-2
8.2	Table descriptions	8-2
8.2.1	gsc_product table—gscpr	8-3
8.2.2	gsc_product_module table—gscpm	8-4
9.	Multi-media and Comment Group	9-1
9.1	Multi-media and Comment group structure	9-2
9.2	Table descriptions	9-2
9.2.1	gsc_multi_media_type table—gscmm	9-3
9.2.2	gsm_comment table—gsmcm	9-4
9.2.3	gsm_multi_media table—gsmmm	9-5

10. Object Group	10-1
10.1 Object group structure	10-2
10.1.1 Main tables in Object group	10-2
10.1.2 Static definition tables	10-4
10.1.3 Class, master, and instance values	10-7
10.2 Managers	10-8
10.3 Table descriptions	10-9
10.3.1 ryc_smartobject table—rycso	10-10
10.3.2 gsc_object_type table—gscot	10-12
10.3.3 ryc_attribute table—rycat	10-15
10.3.4 ryc_attribute_group table—rycap	10-16
10.3.5 ryc_attribute_value table—rycav	10-17
10.3.6 ryc_layout table—rycla	10-19
10.3.7 ryc_object_instance table—rycoi	10-20
10.3.8 ryc_page table—rycpa	10-22
10.3.9 ryc_smartlink table—rycsm	10-23
10.3.10 ryc_smartlink_type table—rycst	10-24
10.3.11 ryc_supported_link table—rycsl	10-25
10.3.12 ryc_ui_event table—rycue	10-26
10.3.13 rym_data_version table—rycdv	10-28
11. Security Group	11-1
11.1 Security group structure	11-2
11.2 Managers	11-2
11.3 Table descriptions	11-3
11.3.1 gsc_security_control table—gscsc	11-4
11.3.2 gsm_field table—gsmff	11-5
11.3.3 gsm_group_allocation table—gscsm	11-6
11.3.4 gsm_range table—gsmra	11-7
11.3.5 gsm_security_structure table—gsmss	11-8
11.3.6 gsm_token table—gsmt0	11-9
12. Sequence Group	12-1
12.1 Sequence group structure	12-2
12.2 Table descriptions	12-2
12.2.1 gsc_sequence table—gscsq	12-2
12.2.2 gsc_next_sequence table—gscsn	12-4

13. Session and Configuration Group	13-1
13.1 Session and Configuration group structure	13-2
13.2 Managers	13-3
13.3 Table descriptions	13-3
13.3.1 gsm_session_type table—gscse	13-4
13.3.2 gsc_logical_service table—gscls	13-5
13.3.3 gsc_manager_type table—gscmt	13-6
13.3.4 gsc_service_type table—gscst	13-8
13.3.5 gsc_session_property table—gscsp	13-9
13.3.6 gsm_physical_service table—gsmpy	13-10
13.3.7 gsm_required_manager table—gsmrm	13-11
13.3.8 gsm_server_context table—gsmrc	13-12
13.3.9 gsm_session_service table—gsmssv	13-13
13.3.10 gsm_session_type_property table—gsmstyp	13-14
13.3.11 gsm_valid_object_partition table—gsmvop	13-15
13.3.12 gst_context_scope table—gstcs	13-15
13.3.13 gst_session table—gstss	13-17
14. Status Group	14-1
14.1 Status group structure	14-2
14.2 Table descriptions	14-2
14.2.1 gsm_category table—gsmca	14-3
14.2.2 gsm_status table—gsmst	14-4
14.2.3 gsm_status_history table—gsmsh	14-6
15. Treeview Group	15-1
15.1 Treeview group structure	15-2
15.2 Managers	15-2
15.3 Table descriptions	15-2
15.3.1 gsm_node table—gsmnd	15-3
15.3.2 rym_wizard_tree table—rymwt	15-4
16. User Group	16-1
16.1 User group structure	16-2
16.2 Managers	16-2
16.3 Table descriptions	16-3
16.3.1 gsm_user table—gsmus	16-4
16.3.2 gsm_user_allocation table—gsmul	16-6
16.3.3 gsm_user_category table—gsmuc	16-8
16.3.4 gst_password_history table—gstph	16-9

17. User Profile Group	17-1
17.1 User Profile group structure	17-2
17.2 Table descriptions	17-2
17.2.1 gsc_profile_code table—gscpc	17-3
17.2.2 gsc_profile_type table—gscpf	17-4
17.2.3 gsm_profile_data table—gsmpf	17-4
18. Other Tables	18-1
18.1 gst_audit table—gstad	18-2
18.1.1 Table description.	18-2
18.2 gsm_help table—gsmhe	18-4
18.2.1 Table description.	18-4
18.3 gsm_login_company table—gsmlg	18-5
18.3.1 Table description.	18-5
19. Legacy and Developing Structures	19-1
19.1 Custom Procedure group	19-2
19.2 Flow group	19-3
19.3 Profile group	19-4
19.4 Reporting Group	19-5
A. Table Names and Acronyms	A-1
A.1 FLA to table name reference	A-2
A.2 Table name to FLA reference	A-8
Index	Index-1

Figures

Figure 1–1:	<i>One to zero, one, or more</i> cardinality	1–2
Figure 1–2:	<i>One to one or more</i> cardinality	1–2
Figure 1–3:	<i>One to zero or one</i> cardinality	1–3
Figure 1–4:	<i>One to exactly (N)</i> cardinality	1–3
Figure 2–1:	Customization group	2–2
Figure 3–1:	Deployment and Versioning group	3–2
Figure 4–1:	Entity and Defaults group	4–3
Figure 5–1:	Error group	5–2
Figure 6–1:	Globalization group	6–2
Figure 7–1:	Menu and Toolbar group	7–3
Figure 8–1:	Module group	8–2
Figure 9–1:	Multi-media and Comment group	9–2
Figure 10–1:	ryc_smartobject table	10–2
Figure 10–2:	Main tables in Object group	10–3
Figure 10–3:	Attribute static definition tables	10–4
Figure 10–4:	SmartLink static definition tables	10–5
Figure 10–5:	Page layout static definition tables	10–6
Figure 10–6:	Table relationships for UI event values	10–7
Figure 10–7:	Table relationships for object attribute values	10–8
Figure 11–1:	Security group	11–2
Figure 12–1:	Sequence group	12–2
Figure 13–1:	Session and Configuration group	13–2
Figure 14–1:	Status group	14–2
Figure 15–1:	Treeview group	15–2
Figure 16–1:	User group	16–2
Figure 17–1:	User Profile group	17–2
Figure 18–1:	gst_audit table	18–2
Figure 18–2:	gsm_help table	18–4
Figure 18–3:	gsm_login_company table	18–5
Figure 19–1:	Custom Procedure group	19–2
Figure 19–2:	Flow group	19–3
Figure 19–3:	Profile group	19–4
Figure 19–4:	Reporting group	19–6

Tables

Table 1–1:	Table purpose codes	1–5
Table 1–2:	Table information codes	1–6
Table 2–1:	ryc_customization_result table information	2–3
Table 2–2:	ryc_customization_result index information	2–4
Table 2–3:	ryc_customization_type table information	2–4
Table 2–4:	ryc_customization_type index information	2–5
Table 2–5:	ryc_render_type table information	2–5
Table 2–6:	ryc_render_type index information	2–6
Table 2–7:	rym_customization table information	2–6
Table 2–8:	rym_customization index information	2–6
Table 3–1:	gst_deployment table information	3–4
Table 3–2:	gst_deployment index information	3–4
Table 3–3:	gsc_dataset_entity table information	3–5
Table 3–4:	gsc_dataset_entity index information	3–6
Table 3–5:	gsc_deploy_dataset table information	3–7
Table 3–6:	gsc_deploy_dataset index information	3–7
Table 3–7:	gsc_deploy_package table information	3–8
Table 3–8:	gsc_deploy_package index information	3–8
Table 3–9:	gsc_package_dataset table information	3–9
Table 3–10:	gsc_package_dataset index information	3–9
Table 3–11:	gsc_scm_tool table information	3–10
Table 3–12:	gsc_scm_tool index information	3–10
Table 3–13:	gsm_release table information	3–11
Table 3–14:	gsm_release index information	3–11
Table 3–15:	gsm_scm_xref table information	3–12
Table 3–16:	gsm_scm_xref index information	3–12
Table 3–17:	gst_dataset_file table information	3–13
Table 3–18:	gst_dataset_file index information	3–13
Table 3–19:	gst_record_version table information	3–14
Table 3–20:	gst_record_version index information	3–15
Table 3–21:	gst_release_version table information	3–16
Table 3–22:	gst_release_version index information	3–16
Table 3–23:	ryt_dbupdate_status table information	3–17
Table 3–24:	ryt_dbupdate_status index information	3–17
Table 4–1:	gsc_entity_mnemonic table information	4–5
Table 4–2:	gsc_entity_mnemonic index information	4–6
Table 4–3:	gsc_data_tag table information	4–6
Table 4–4:	gsc_data_tag index information	4–7
Table 4–5:	gsc_default_code table information	4–7
Table 4–6:	gsc_default_code index information	4–8
Table 4–7:	gsc_default_set table information	4–8
Table 4–8:	gsc_default_set index information	4–8
Table 4–9:	gsc_default_set_usage table information	4–9

Table 4-10:	gsc_default_set_usage index information	4-9
Table 4-11:	gsc_entity_display_field table information	4-10
Table 4-12:	gsc_entity_display_field index information	4-10
Table 4-13:	gsm_control_code table information	4-11
Table 4-14:	gsm_control_code index information	4-11
Table 4-15:	gsm_entity_field table information	4-12
Table 4-16:	gsm_entity_field index information	4-12
Table 4-17:	gsm_entity_field_value table information	4-12
Table 4-18:	gsm_entity_field_value index information	4-13
Table 4-19:	gsm_external_xref table information	4-13
Table 4-20:	gsm_external_xref index information	4-14
Table 4-21:	gsm_filter_data table information	4-15
Table 4-22:	gsm_filter_data index information	4-16
Table 4-23:	gsm_filter_set table information	4-16
Table 4-24:	gsm_filter_set index information	4-17
Table 4-25:	gsm_tagged_data table information	4-17
Table 4-26:	gsm_tagged_data index information	4-18
Table 4-27:	ryc_relationship table information	4-18
Table 4-28:	ryc_relationship index information	4-19
Table 4-29:	ryc_relationship_field table information	4-20
Table 4-30:	ryc_relationship_field index information	4-20
Table 4-31:	ryc_ri_default table information	4-21
Table 4-32:	ryc_ri_default index information	4-21
Table 5-1:	gsc_error table information	5-3
Table 5-2:	gsc_error index information	5-3
Table 5-3:	gst_error_log table information	5-4
Table 5-4:	gst_error_log index information	5-5
Table 6-1:	gsc_language table information	6-4
Table 6-2:	gsc_language index information	6-4
Table 6-3:	gsc_global_control table information	6-5
Table 6-4:	gsc_global_control index information	6-5
Table 6-5:	gsc_global_default table information	6-6
Table 6-6:	gsc_global_default index information	6-6
Table 6-7:	gsc_language_text table information	6-7
Table 6-8:	gsc_language_text index information	6-7
Table 6-9:	gsc_nationality table information	6-8
Table 6-10:	gsc_nationality index information	6-8
Table 6-11:	gsm_country table information	6-9
Table 6-12:	gsm_country index information	6-9
Table 6-13:	gsm_currency table information	6-10
Table 6-14:	gsm_currency index information	6-10
Table 6-15:	gsm_translation table information	6-11
Table 6-16:	gsm_translation index information	6-12
Table 7-1:	gsm_menu_item table information	7-5
Table 7-2:	gsm_menu_item index information	7-6

Table 7-3:	gsc_instance_attribute table information	7-7
Table 7-4:	gsc_instance_attribute index information	7-7
Table 7-5:	gsc_item_category table information	7-8
Table 7-6:	gsc_item_category index information	7-8
Table 7-7:	gsm_menu_structure table information	7-9
Table 7-8:	gsm_menu_structure index information	7-9
Table 7-9:	gsm_menu_structure_item table information	7-10
Table 7-10:	gsm_menu_structure_item index information	7-10
Table 7-11:	gsm_object_menu_structure table information	7-11
Table 7-12:	gsm_object_menu_structure index information	7-11
Table 7-13:	gsm_toolbar_menu_structure table information	7-12
Table 7-14:	gsm_toolbar_menu_structure index information	7-12
Table 7-15:	gsm_translated_menu_item table information	7-13
Table 7-16:	gsm_translated_menu_item index information	7-13
Table 8-1:	gsc_product table information	8-3
Table 8-2:	gsc_product index information	8-3
Table 8-3:	gsc_product_module table information	8-4
Table 8-4:	gsc_product_module index information	8-4
Table 9-1:	gsc_multi_media_type table information	9-3
Table 9-2:	gsc_multi_media_type index information	9-3
Table 9-3:	gsm_comment table information	9-4
Table 9-4:	gsm_comment index information	9-5
Table 9-5:	gsm_multi_media table information	9-5
Table 9-6:	gsm_multi_media index information	9-6
Table 10-1:	ryc_smartobject table information	10-10
Table 10-2:	ryc_smartobject index information	10-11
Table 10-3:	gsc_object_type table information	10-13
Table 10-4:	gsc_object_type index information	10-13
Table 10-5:	ryc_attribute table information	10-15
Table 10-6:	ryc_attribute index information	10-15
Table 10-7:	ryc_attribute_group table information	10-16
Table 10-8:	ryc_attribute_group index information	10-16
Table 10-9:	ryc_attribute_value table information	10-18
Table 10-10:	ryc_attribute_value index information	10-19
Table 10-11:	ryc_layout table information	10-20
Table 10-12:	ryc_layout index information	10-20
Table 10-13:	ryc_object_instance table information	10-21
Table 10-14:	ryc_object_instance index information	10-21
Table 10-15:	ryc_page table information	10-22
Table 10-16:	ryc_page index information	10-22
Table 10-17:	ryc_smartlink table information	10-23
Table 10-18:	ryc_smartlink index information	10-24
Table 10-19:	ryc_smartlink_type table information	10-24
Table 10-20:	ryc_smartlink_type index information	10-25
Table 10-21:	ryc_supported_link table information	10-25

Table 10–22:	ryc_supported_link index information	10–26
Table 10–23:	ryc_ui_event table information	10–26
Table 10–24:	ryc_ui_event index information	10–27
Table 10–25:	rym_data_version table information	10–28
Table 10–26:	rym_data_version index information	10–28
Table 11–1:	gsc_security_control table information	11–4
Table 11–2:	gsc_security_control index information	11–4
Table 11–3:	gsm_field table information	11–5
Table 11–4:	gsm_field index information	11–5
Table 11–5:	gsm_group_allocation table information	11–6
Table 11–6:	gsm_group_allocation index information	11–6
Table 11–7:	gsm_range table information	11–7
Table 11–8:	gsm_range index information	11–7
Table 11–9:	gsm_security_structure table information	11–8
Table 11–10:	gsm_security_structure index information	11–8
Table 11–11:	gsm_token table information	11–9
Table 11–12:	gsm_token index information	11–9
Table 12–1:	gsc_sequence table information	12–3
Table 12–2:	gsc_sequence index information	12–3
Table 12–3:	gsc_next_sequence table information	12–4
Table 12–4:	gsc_next_sequence index information	12–4
Table 13–1:	gsm_session_type table information	13–4
Table 13–2:	gsm_session_type index information	13–4
Table 13–3:	gsc_logical_service table information	13–5
Table 13–4:	gsc_logical_service index information	13–6
Table 13–5:	gsc_manager_type table information	13–6
Table 13–6:	gsc_manager_type index information	13–7
Table 13–7:	gsc_service_type table information	13–8
Table 13–8:	gsc_service_type index information	13–8
Table 13–9:	gsc_session_property table information	13–9
Table 13–10:	gsc_session_property index information	13–9
Table 13–11:	gsm_physical_service table information	13–10
Table 13–12:	gsm_physical_service index information	13–10
Table 13–13:	gsm_required_manager table information	13–11
Table 13–14:	gsm_required_manager index information	13–11
Table 13–15:	gsm_server_context table information	13–12
Table 13–16:	gsm_server_context index information	13–12
Table 13–17:	gsm_session_service table information	13–13
Table 13–18:	gsm_session_service index information	13–13
Table 13–19:	gsm_session_type_property table information	13–14
Table 13–20:	gsm_session_type_property index information	13–14
Table 13–21:	gsm_valid_object_partition table information	13–15
Table 13–22:	gsm_valid_object_partition index information	13–15
Table 13–23:	gst_context_scope table information	13–16
Table 13–24:	gst_context_scope index information	13–17

Table 13–25:	gst_session table information	13–17
Table 13–26:	gst_session index information	13–18
Table 14–1:	gsm_category table information	14–3
Table 14–2:	gsm_category index information	14–4
Table 14–3:	gsm_status table information	14–5
Table 14–4:	gsm_status index information	14–5
Table 14–5:	gsm_status_history table information	14–6
Table 14–6:	gsm_status_history index information	14–7
Table 15–1:	gsm_node table information	15–3
Table 15–2:	gsm_node index information	15–4
Table 15–3:	rym_wizard_tree table information	15–5
Table 15–4:	rym_wizard_tree index information	15–6
Table 16–1:	gsm_user table information	16–4
Table 16–2:	gsm_user index information	16–5
Table 16–3:	gsm_user_allocation table information	16–6
Table 16–4:	gsm_user_allocation index information	16–7
Table 16–5:	gsm_user_category table information	16–8
Table 16–6:	gsm_user_category index information	16–9
Table 16–7:	gst_password_history table information	16–9
Table 16–8:	gst_password_history index information	16–10
Table 17–1:	gsc_profile_code table information	17–3
Table 17–2:	gsc_profile_code index information	17–3
Table 17–3:	gsc_profile_type table information	17–4
Table 17–4:	gsc_profile_type index information	17–4
Table 17–5:	gsm_profile_data table information	17–5
Table 17–6:	gsm_profile_data index information	17–5
Table 18–1:	gst_audit table information	18–2
Table 18–2:	gst_audit index information	18–3
Table 18–3:	gsm_help table information	18–4
Table 18–4:	gsm_help index information	18–5
Table 18–5:	gsm_login_company table information	18–6
Table 18–6:	gsm_login_company index information	18–6
Table A–1:	FLA to table name reference	A–2
Table A–2:	Table name to FLA reference	A–8

Preface

Purpose

Progress Dynamics® is a repository-based framework and application development environment (ADE) designed to accelerate the building and deployment of distributed enterprise applications. It offers you the ability to rapidly build, deploy, and customize competitive business applications using the Progress® OpenEdge™ environment. By following the Progress Dynamics' prescriptive approach, you can decrease the time required to bring your business application to deployment.

Audience

This reference is intended for developers who design applications with Progress Dynamics.

Organization of this manual

This manual is organized into chapters that outline groups of tables that serve a common purpose for the Progress Dynamics framework. The chapters are as follows:

[Chapter 1, “The Progress Dynamics Repository”](#)

Provides basic information on the design of the Repository.

[Chapter 2, “Customization Group”](#)

Describes the tables that store customization information.

[Chapter 3, “Deployment and Versioning Group”](#)

Describes the tables that store deployment and versioning information.

[Chapter 4, “Entity and Defaults Group”](#)

Describes the tables that store information about entities and defaults.

[Chapter 5, “Error Group”](#)

Describes the tables that store error message information.

[Chapter 6, “Globalization Group”](#)

Describes the tables that store information used to globalize a Progress Dynamics application.

[Chapter 7, “Menu and Toolbar Group”](#)

Describes the tables that store information about menus and toolbars.

[Chapter 8, “Module Group”](#)

Describes the tables that store information about modules

[Chapter 9, “Multi-media and Comment Group”](#)

Describes the tables that store information about multi-media files and comments used in a Progress Dynamics application.

[Chapter 10, “Object Group”](#)

Describes the tables that store information about the objects in a Progress Dynamics application.

[Chapter 11, “Security Group”](#)

Describes the tables that store information about security allocations.

[Chapter 12, “Sequence Group”](#)

Describes the tables that store information about Progress Dynamics-specific sequences.

[Chapter 13, “Session and Configuration Group”](#)

Describes the tables that store session and configuration information.

[Chapter 14, “Status Group”](#)

Describes the tables that store status information.

Chapter 15, “Treeview Group”

Describes the tables that store information used by treeviews.

Chapter 16, “User Group”

Describes the tables that store information about individual users.

Chapter 17, “User Profile Group”

Describes the tables that support the use of user profiles.

Chapter 18, “Other Tables”

Describes several tables that do not fit into any group.

Chapter 19, “Legacy and Developing Structures”

Describes several groups of tables that are included in the Repository to support legacy functionality or to aid possible developments in the Progress Dynamics framework.

Appendix A, “Table Names and Acronyms”

Provides a quick reference for finding the Repository table name for an FLA, or the FLA for a Repository table name.

How to use this manual

This reference is intended to give you a broad view of the structure of the Progress Dynamics Repository. More specific information about table and field properties can be accessed through the Data Administration and Data Dictionary tools.

NOTE: This edition of the *Progress Dynamics Repository Reference* is based on the Repository structure including all changes up through those in `icfdb020027delta.df`. If you have upgraded your Repository to a version that applied higher numbered delta files, check for changes made in those files.

Typographical conventions

This manual uses the following typographical conventions:

- **Bold typeface** indicates:
 - Commands or characters that the user types
 - That a word carries particular weight or emphasis
 - Names of user interface elements
- *Italic typeface* indicates:
 - Progress variable information that the user supplies
 - New terms
 - Titles of complete publications
- Monospaced typeface indicates:
 - Code examples
 - System output
 - Operating system filenames and pathnames

The following typographical conventions are used to represent keystrokes:

- Small capitals are used for Progress key functions and generic keyboard keys.
END-ERROR, GET, GO
ALT, CTRL, SPACEBAR, TAB
- When you have to press a combination of keys, they are joined by a hyphen. You press and hold down the first key, then press the second key.
CTRL-X
- When you have to press and release one key, then press another key, the key names are separated with a space.
ESCAPE H
ESCAPE CURSOR-LEFT

Syntax notation

The syntax for each component follows a set of conventions:

- Uppercase words are keywords. Although they are always shown in uppercase, you can use either uppercase or lowercase when using them in a procedure.

In this example, `ACCUM` is a keyword:

Syntax

```
ACCUM aggregate expression
```

- Italics identify options or arguments that you must supply. These options can be defined as part of the syntax or in a separate syntax identified by the name in italics. In the `ACCUM` function above, the *aggregate* and *expression* options are defined with the syntax for the `ACCUM` function in the [Progress Language Reference](#).
- You must end all statements (except for `DO`, `FOR`, `FUNCTION`, `PROCEDURE`, and `REPEAT`) with a period. `DO`, `FOR`, `FUNCTION`, `PROCEDURE`, and `REPEAT` statements can end with either a period or a colon, as in this example:

```
FOR EACH Customer:
  DISPLAY Name.
END.
```

- Square brackets (`[]`) around an item indicate that the item, or a choice of one of the enclosed items, is optional.

In this example, `STREAM stream`, `UNLESS-HIDDEN`, and `NO-ERROR` are optional:

Syntax

```
DISPLAY [ STREAM stream ] [ UNLESS-HIDDEN ] [ NO-ERROR ]
```

In some instances, square brackets are not a syntax notation, but part of the language.

For example, this syntax for the INITIAL option uses brackets to bound an initial value list for an array variable definition. In these cases, normal text brackets ([]) are used:

Syntax

```
INITIAL [ constant [ , constant ] . . . ]
```

NOTE: The ellipsis (. . .) indicates repetition, as shown in a following description.

- Braces ({ }) around an item indicate that the item, or a choice of one of the enclosed items, is required.

In this example, you must specify the items BY and *expression* and can optionally specify the item DESCENDING, in that order:

Syntax

```
{ BY expression [ DESCENDING ] }
```

In some cases, braces are not a syntax notation, but part of the language.

For example, a called external procedure must use braces when referencing arguments passed by a calling procedure. In these cases, normal text braces ({ }) are used:

Syntax

```
{ &argument-name }
```

- A vertical bar (|) indicates a choice.

In this example, EACH, FIRST, and LAST are optional, but you can only choose one:

Syntax

```
PRESELECT [ EACH | FIRST | LAST ] record-phrase
```

In this example, you must select one of *logical-name* or *alias*:

Syntax

```
CONNECTED ( { logical-name | alias } )
```

- Ellipses (. . .) indicate that you can choose one or more of the preceding items. If a group of items is enclosed in braces and followed by ellipses, you must choose one or more of those items. If a group of items is enclosed in brackets and followed by ellipses, you can optionally choose one or more of those items.

In this example, you must include two expressions, but you can optionally include more. Note that each subsequent expression must be preceded by a comma:

Syntax

```
MAXIMUM ( expression , expression [ , expression ] . . . )
```

In this example, you must specify MESSAGE, then at least one of *expression* or SKIP, but any additional number of *expression* or SKIP is allowed:

Syntax

```
MESSAGE { expression | SKIP [ ( n ) ] } . . .
```

In this example, you must specify {*include-file*, then optionally any number of *argument* or *&argument-name* = "*argument-value*", and then terminate with }:

Syntax

```
{ include-file  
  [ argument | &argument-name = "argument-value" ] . . . }
```

- In some examples, the syntax is too long to place in one horizontal row. In such cases, **optional** items appear individually bracketed in multiple rows in order, left-to-right and top-to-bottom. This order generally applies, unless otherwise specified. **Required** items also appear on multiple rows in the required order, left-to-right and top-to-bottom. In cases where grouping and order might otherwise be ambiguous, braced (required) or bracketed (optional) groups clarify the groupings.

In this example, WITH is followed by several optional items:

Syntax

```
WITH [ ACCUM max-length ] [ expression DOWN ]  
    [ CENTERED ] [ n COLUMNS ] [ SIDE-LABELS ]  
    [ STREAM-IO ]
```

In this example, ASSIGN requires one of two choices: either one or more of *field*, or one of *record*. Other options available with either *field* or *record* are grouped with braces and brackets. The open and close braces indicate the required order of options:

Syntax

```
ASSIGN { { [ FRAME frame ]  
          { field [ = expression ] }  
          [ WHEN expression ]  
        } ...  
      | { record [ EXCEPT field ... ] }  
    }
```

Progress messages

Progress displays several types of messages to inform you of routine and unusual occurrences:

- Execution messages inform you of errors encountered while Progress is running a procedure (for example, if Progress cannot find a record with a specified index field value).
- Compile messages inform you of errors found while Progress is reading and analyzing a procedure prior to running it (for example, if a procedure references a table name that is not defined in the database).
- Startup messages inform you of unusual conditions detected while Progress is getting ready to execute (for example, if you entered an invalid startup parameter).

After displaying a message, Progress proceeds in one of several ways:

- Continues execution, subject to the error-processing actions that you specify, or that are assumed, as part of the procedure. This is the most common action taken following execution messages.
- Returns to the Progress Procedure Editor so that you can correct an error in a procedure. This is the usual action taken following compiler messages.
- Halts processing of a procedure and returns immediately to the Procedure Editor. This does not happen often.
- Terminates the current session.

Progress messages end with a message number in parentheses. In this example, the message number is 200:

```
** Unknown table name table. (200)
```

Use Progress online help to get more information about Progress messages. Many Progress tools include the following Help menu options to provide information about messages:

- Choose **Help—Recent Messages** to display detailed descriptions of the most recent Progress message and all other messages returned in the current session.
- Choose **Help—Messages**, then enter the message number to display a description of any Progress message. (If you encounter an error that terminates Progress, make a note of the message number before restarting.)
- In the Procedure Editor, press the **HELP** key (**F2** or **CTRL-W**).

The Progress Dynamics Repository

This chapter provides a background for understanding the material on the Progress Dynamics® Repository presented in the following chapters. This chapter covers the following topics:

- [Repository model](#)
- [Repository conventions](#)

1.1 Repository model

The Repository for Progress Dynamics Version 2 was designed with the ERwin database modeling tool. The model, `icfdb.er1`, and its template, `icftemplate.ert`, ship with Progress Dynamics. You can locate these files in `DLC\src\icf\db\icf\doc`. You need ERwin Version 4.1 SP2 (Build 4.1.2771) to open these files.

The table figures used in later chapters are taken from the Progress Dynamics Repository model. The figures use the Information Engineering (IE) notation form. These figures show the way various cardinalities are depicted in IE notation.

Figure 1–1 shows the notation for “One to zero, one, or more” cardinality.

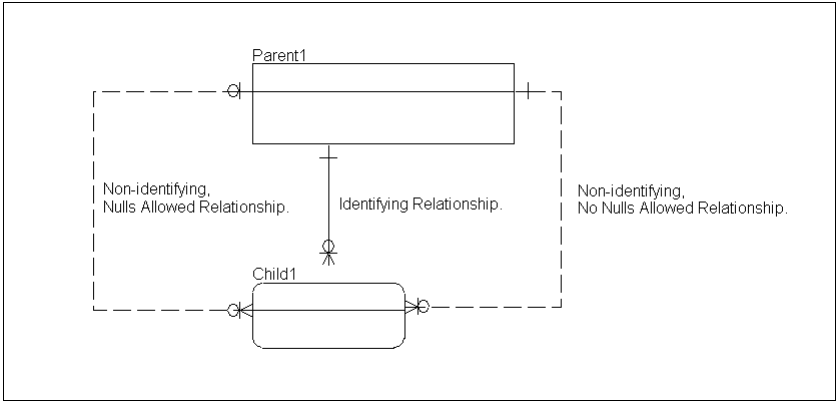


Figure 1–1: **One to zero, one, or more cardinality**

Figure 1–2 shows the notation for “One to one or more” cardinality.

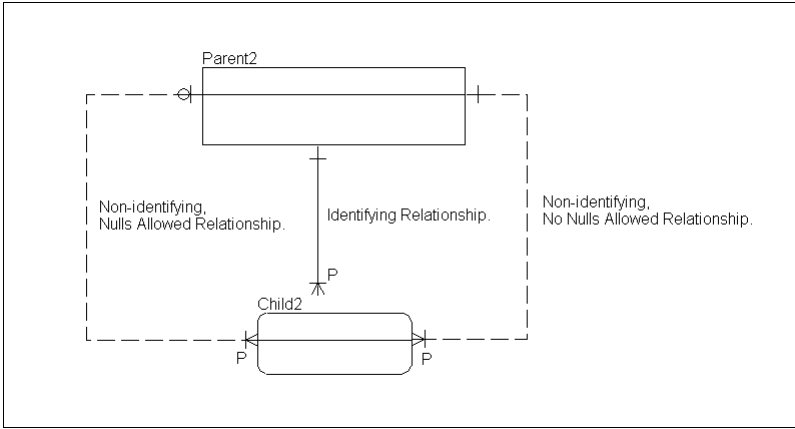


Figure 1–2: **One to one or more cardinality**

Figure 1–3 shows the notation for “One to zero or one” cardinality.

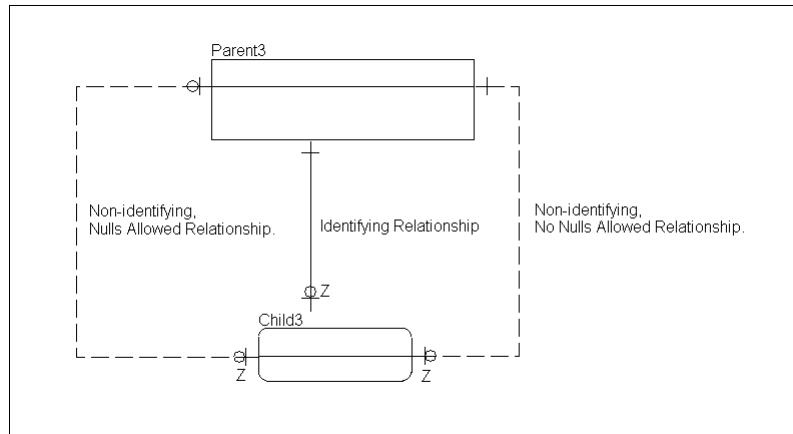


Figure 1–3: One to zero or one cardinality

Figure 1–4 shows the notation for “One to exactly (N)” cardinality.

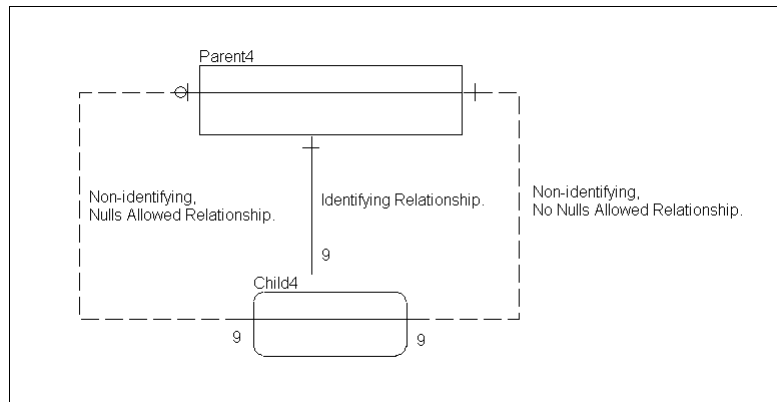


Figure 1–4: One to exactly (N) cardinality

1.2 Repository conventions

The Progress Dynamics Repository makes use of several conventions that aid in efficient design of Progress Dynamics applications. The following sections discuss the Repository conventions.

For more information on using the Repository, see the chapter on database design principles in the *Progress Dynamics Developer's Guide* and the chapters on creating a new manager and understanding the Repository's object tables in the *Progress Dynamics Programming Handbook*.

1.2.1 Schema definition files

The Progress Dynamics Repository's schema changes to support new functionality with each release. Each release of Progress Dynamics ships with several definition (.df) files for the Repository in the `src\icf\db\icf\dfd` directory. This directory contains a baseline definition file and several incremental definition files.

The baseline definition file is named `icfdbfull.df`. This is the definition file to use if you are building a new Repository starting with an empty Progress database.

The incremental definition files are named using the convention, `{dbname}{version}{type}.df`, where:

`{dbname}` is the name of database, ICFDB in this case.

`{version}` is the sequential version number of the database, incremented with each change. The version is a six digit number assigned from the Repository's `seq_ICFDB_DBVersion` sequence.

`{type}` indicates the type of definition file as follows:

- **erwin** — Straight dump from the ERwin model. This is only used for database administration. It is not used for deployment.
- **delta** — Incremental schema changes from the previous `icfdbfull.df`.

NOTE: This edition of the *Progress Dynamics Repository Reference* is based on the Repository structure including all changes up through those in `icfdb020027delta.df`. If you have upgraded your Repository to a version that applied higher numbered delta files, check for changes made in those files.

1.2.2 Table conventions

Tables in the Repository use the following conventions:

- Table names should be unique.
- Table names should be meaningful and preferably not abbreviated.
- The entire table name cannot exceed 29 characters.
- The valid characters for table names are lowercase “a . . z” and the underscore (_).
- The first four characters of each table name are a standard prefix, constructed as follows:
 - a) The first two characters designate the general purpose of the table. There are currently two valid designations, as described in [Table 1–1](#).

Table 1–1: Table purpose codes

Designation	Description
gs	System data that controls the environment, such as objects, menu items, messages, managers, services, and flows Or Application data, such as users, security, auditing, profiles, categories, and comments
ry	Repository data, such as objects, attributes, links, pages, bands, and actions

- b) The third character designates the general type of information stored in the table. [Table 1–2](#) describes the valid designations.

Table 1–2: Table information codes

Designation	Description
c	Static control and parameter information
m	Master information that is likely to change
r	Raw data, unformatted and unprocessed
t	Transaction information

- c) The fourth character is an underscore (_).
- Each tables has an unique acronym called the FLA. The FLA is the table name’s three-character prefix and two mnemonic characters for the rest of the table name. For example, the FLA for the `gsm_user` table is `gsmus`. The FLA can be used as a substitute for the full table name in your code. The Progress Dynamics framework also supports the use of the eight-character table dump name for this purpose.
 - Each table should have a field whose name is the table name, without its prefix, and the suffix, “_obj”. For example, the `gsm_user` table has a field named `user_obj`. Unless the table has no child table, this field should be the primary key for the table. The field has a domain of “o_obj” and is never displayed to users. It is used primarily for relationships between tables.

1.2.3 Field conventions

Fields in the Repository use the following conventions:

- Field name should be between five and 29 characters long.
- The valid characters for field names are lowercase “a . . z” and the underscore (_).
- Field names should be as specific as possible. For example, use “customer_name” rather than “name”.
- Fields in different tables that can be used as join fields should have the same name.

Customization Group

This chapter covers the group of tables in the Progress Dynamics Repository that store information about customizations of objects in your application. This chapter covers the following topics:

- [Customization group structure](#)
- [Managers](#)
- [Table descriptions](#)

2.1 Customization group structure

The Customization group stores information about customizations of your application. [Figure 2-1](#) shows the structure and relationships of the tables in this group.

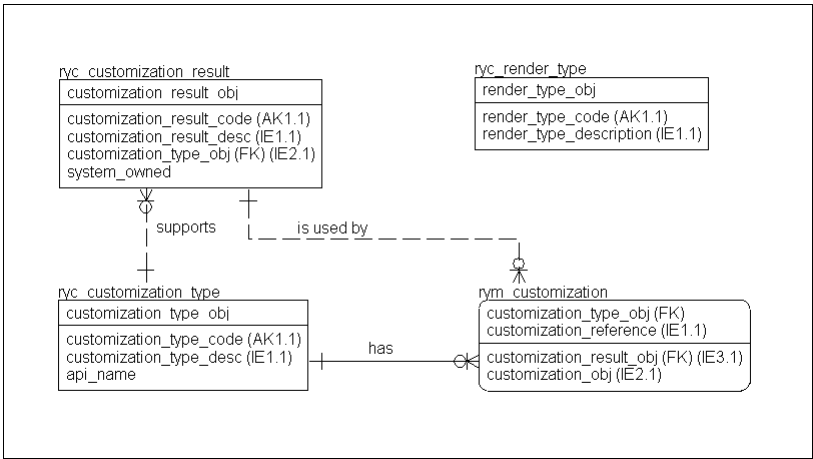


Figure 2-1: Customization group

Customizations are special purpose changes to the normal way an application runs. For example, you might want to customize some elements of your application's user interface (UI) when it runs in a Web browser. Or, you might want to change your application's behavior for certain classes of users.

2.2 Managers

The Customization and Repository Managers use data from these tables to apply customizations to user interfaces. For more detail about the temp-tables and APIs these Managers use to manipulate data from the Customization group tables, see the chapters on the Customization and Repository Managers in the [Progress Dynamics Managers API Reference](#).

2.3 Table descriptions

The following sections provide detailed information about each table in the Customization group. The Customization group contains the following database tables:

- [ryc_customization_result table—ryccr](#)
- [ryc_customization_type table—ryccy](#)
- [ryc_render_type table—rycrt](#)
- [rym_customization table—rymcz](#)

2.3.1 ryc_customization_result table—ryccr

This table stores the possible customization codes describing levels of customization. Many types of customization exist, including user interface (UI) type customizations (such as HTML, DHTML, GUI), user category customizations, user customizations, and company level customizations. The types of customizations are defined in the `ryc_customization_type` table. To avoid confusion and to enable some control of result code usage, each result code must be for a specific customization type.

This table contains a code and description field. To avoid conflicting uses of result codes, use meaningful codes and descriptions. To identify for which type of customization the result code is used, the table is joined to the `ryc_customization_type` table.

Examples of meaningful result codes are using user login names or job titles for user level customizations. You might use a UI's acronym as the result code for that UI type customization. You might use a language's name as the result code for that language customization.

This method provides maximum flexibility to the required level of customization. The customization types define the levels of supported customization. The result codes define the possible values for each customization type.

[Table 2–1](#) lists the table's FLA, fields, and foreign keys.

Table 2–1: ryc_customization_result table information

Table FLA	Fields (data type)	Foreign keys
ryccr	customization_result_obj (Decimal) customization_result_code (Character) customization_result_desc (Character) customization_type_obj (Decimal) system_owned (Logical)	customization_result_obj customization_type_obj

Table 2–2 gives details of the table’s indexes.

Table 2–2: ryc_customization_result index information

Index name	Elements	Type
XPKryc_customization_result	customization_result_obj	Primary Unique
XAK1ryc_customization_result	customization_result_code	Unique
XIE1ryc_customization_result	customization_result_desc	Nonunique
XIE2ryc_customization_result	customization_type_obj	Nonunique

Certain result codes, such as supported UI type customizations, are provided as part of the framework. These result codes are defined with a system-owned flag set to YES. Maintenance of these codes is restricted to users authorized to maintain system data.

When defining these framework-supplied result codes, be careful to avoid potential conflicts with result codes used by Progress Dynamics applications.

2.3.2 ryc_customization_type table—ryccy

This table lists the customizations defined for your application and the API functions that determine the value of the customization at runtime.

Table 2–3 lists the table’s FLA, fields, and foreign keys.

Table 2–3: ryc_customization_type table information

Table FLA	Fields (data type)	Foreign keys
ryccy	customization_type_obj (Decimal) customization_type_code (Character) customization_type_desc (Character) api_name (Character)	customization_type_obj

Table 2–4 gives details of the table’s indexes.

Table 2–4: ryc_customization_type index information

Index name	Elements	Type
XPKryc_customization_type	customization_type_obj	Primary Unique
XAK1ryc_customization_type	customization_type_code	Unique
XIE1ryc_customization_type	customization_type_desc	Nonunique

2.3.3 ryc_render_type table—rycrt

This table defines the supported rendering engines, for example, WEB, GUI, HTML, B2C, .NET, and XML. The table is joined into the gst_session table to identify which rendering engine is active for a session. The main purpose of this table is to optionally join into the ryc_attribute_value and ryc_ui_event tables to enable overriding attribute values and events for different rendering engines. Different overrides might be made at the class, master, and instance level. The render type offers another dimension of customization capability specifically for rendering engines. This table also supports the ability to identify certain attributes and events that are only applicable to specific rendering engines and the ability to override the values of common events and attributes across rendering engines.

Table 2–5 lists the table’s FLA, fields, and foreign keys.

Table 2–5: ryc_render_type table information

Table FLA	Fields (data type)	Foreign keys
rycrt	render_type_obj (Decimal) render_type_code (Character) render_type_description (Character)	render_type_obj

Table 2–6 gives details of the table’s indexes.

Table 2–6: ryc_render_type index information

Index name	Elements	Type
XPKryc_render_type	render_type_obj	Primary Unique
XAK1ryc_render_type	render_type_obj	Unique
XIE1ryc_render_type	render_type_description	Nonunique

2.3.4 rym_customization table—rymcz

This table captures the actual customization results for supported customization types, as defined in the ryc_customization_type table.

Table 2–7 lists the table’s FLA, fields, and foreign keys.

Table 2–7: rym_customization table information

Table FLA	Fields (data type)	Foreign keys
rymcz	customization_type_obj (Decimal) customization_reference (Character) customization_result_obj (Decimal) customization_obj (Decimal)	customization_result_obj customization_type_obj

Table 2–8 gives details of the table’s indexes.

Table 2–8: rym_customization index information

Index name	Elements	Type
XPKrym_customization	customization_type_obj customization_reference	Primary Unique
XIE1rym_customization	customization_reference	Nonunique
XIE2rym_customization	customization_obj	Nonunique
XIE3rym_customization	customization_result_obj	Nonunique

A mandatory join from the `ryc_customization_type` table defines each customization type. The `customization_reference` field is part of the unique key that stores customization values according to their types. For a user level customization, the `customization_reference` field contains a specific user's login code. For a UI type customization, then the field contains the value of a UI type, such as HTML or GUI.

There is a difference between the data in this table and the `ryc_customization_result` table. The `ryc_customization_result` table lists valid values for the result codes. This table stores actual values that can be checked at run time with the appropriate API. The values on this table might differ from the result code values on `ryc_customization_result`. The result code values are reusable.

For example, one customization type is user customization. On this table, there might be a record for each user with a reference pointing at each user's login name. The API specified for the customization type runs to find the current user and looks up the specified result code for the user in this table. There might be several users whose result code value is "manager" and others whose result code value is "engineer." Note the possible reuse of result codes and the difference between the result code values and the customization references.

This table stores the result for the customization through a join to the `ryc_customization_result` table, identifying the result code to use for this specific customization. These tables provide maximum flexibility for customization possibilities. The result codes for a session are evaluated after authentication and then made available to the session.

When reading from tables that support customization, such as `ryc_smartobject` and its related tables, if any matching customizations exist, the appropriate record is read with a matching result code of the highest priority level. Otherwise, the default record is used.

Deployment and Versioning Group

This chapter covers the group of tables in the Progress Dynamics Repository that store information about deployment and versioning. This chapter covers the following topics:

- [Deployment and Versioning group structure](#)
- [Managers](#)
- [Table descriptions](#)

3.1 Deployment and Versioning group structure

The Deployment and Versioning group is used in deploying your applications and data. [Figure 3–1](#) shows the structure and relationships of the tables in this group.

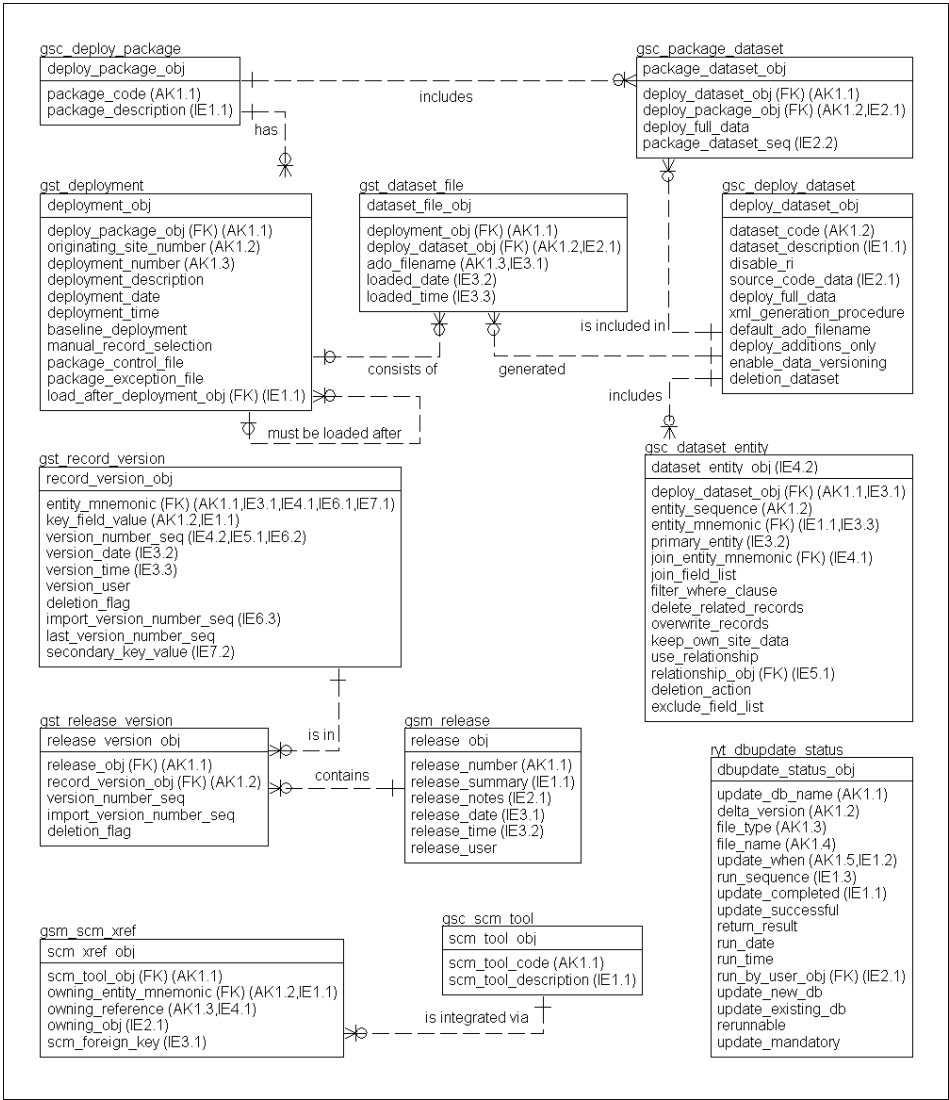


Figure 3–1: Deployment and Versioning group

3.2 Managers

The Referential Integrity Manager uses data from these tables to support data versioning and the reuse of unique keys. For more detail about the temp-tables and APIs the Referential Integrity Manager uses to manipulate data from the Deployment and Versioning group tables, see the chapter on the Referential Integrity Manager in the *Progress Dynamics Managers API Reference*.

3.3 Table descriptions

The following sections provide detailed information about each table in the Deployment and Versioning group. The Deployment and Versioning group contains the following tables:

- [gst_deployment table—gstdp](#)
- [gsc_dataset_entity table—gscde](#)
- [gsc_deploy_dataset table—gscdd](#)
- [gsc_deploy_package table—gscdp](#)
- [gsc_package_dataset table—gscpd](#)
- [gsc_scm_tool table—gscsm](#)
- [gsm_release table—gsmrl](#)
- [gsm_scm_xref table—gsmsx](#)
- [gst_dataset_file table—gstdf](#)
- [gst_record_version table—gstrv](#)
- [gst_release_version table—gstrl](#)
- [ryt_dbupdate_status table—rytds](#)

3.3.1 **gst_deployment table—gstdp**

The `gst_deployment` table is the central table of the Deployment and Versioning group. This table defines an instance of a deployment package, from a particular site. Records in this table might be manually created for the current site, or might be imported as part of loading a deployment package from an external site.

[Table 3–1](#) lists the table’s FLA, fields, and foreign keys.

Table 3–1: `gst_deployment` table information

Table FLA	Fields (data type)	Foreign keys
gstdp	deployment_obj (Decimal) deploy_package_obj (Decimal) originating_site_number (Integer) deployment_number (Integer) deployment_description (Character) deployment_date (Date) deployment_time (Integer) baseline_deployment (Logical) manual_record_selection (Logical) package_control_file (Character) package_exception_file (Character) load_after_deployment_obj (Decimal)	deployment_obj deploy_package_obj

[Table 3–2](#) gives details of the table’s indexes.

Table 3–2: `gst_deployment` index information

Index name	Elements	Type
XPKgst_deployment	deployment_obj	Primary Unique
XAK1gst_deployment	deploy_package_obj originating_site_number deployment_number	Unique
XIE1gst_deployment	load_after_deployment_obj	Nonunique

3.3.2 gsc_dataset_entity table—gscde

This table lists the tables that need to be deployed with the dataset. Each dataset must have a designated primary table, in other words, the main table in the dataset. The join information between the tables must also be specified.

The data in this table can be filtered using the WHERE clause stored in the filter_where_clause field.

Table 3–3 lists the table’s FLA, fields, and foreign keys.

Table 3–3: gsc_dataset_entity table information

Table FLA	Fields (data type)	Foreign keys
gscde	dataset_entity_obj (Decimal) deploy_dataset_obj (Decimal) entity_sequence (Integer) entity_mnemonic (Character) primary_entity (Logical) join_entity_mnemonic (Character) join_field_list (Character) filter_where_clause (Character) delete_related_records (Logical) overwrite_records (Logical) keep_own_site_data (Logical) use_relationship (Logical) relationship_obj (Decimal) deletion_action (Character) exclude_field_list (Character)	deploy_dataset_obj entity_mnemonic relationship_obj

Table 3–4 gives details of the table’s indexes.

Table 3–4: gsc_dataset_entity index information

Index name	Elements	Type
XPkgsc_dataset_entity	dataset_entity_obj	Primary Unique
XAK1gsc_dataset_entity	deploy_dataset_obj entity_sequence	Unique
XIE1gsc_dataset_entity	entity_mnemonic	Nonunique
XIE3gsc_dataset_entity	deploy_dataset_obj primary_entity entity_mnemonic	Nonunique
XIE4gsc_dataset_entity	join_entity_mnemonic dataset_entity_obj	Nonunique
XIE5gsc_dataset_entity	relationship_obj	Nonunique

3.3.3 gsc_deploy_dataset table—gscdd

This table defines the sets of data that need to be deployed to end-user sites and migrated to different workspace databases. Usually it is static data that needs to be deployed. This table and its child table, gsc_dataset_entity, identify which pieces of data are dependent on each other and must be deployed as a set. For example, in order to deploy menu items, the objects on the menu item also need to be deployed.

This table and its child table, gsc_dataset_entity, also define the dataset for deployment of logical objects managed by the source code management tool, such as the ryc_smartobject and related tables.

To deploy and load the data, XML files are generated for the dataset.

The dataset always has a main table and all the related tables that need to be deployed with it, together with the appropriate join information.

Example datasets could be for Progress SmartObjects™, menus, and objects. Most data tables could be defined as separate datasets that include a parent dataset. The parent dataset includes whichever datasets need to be deployed together.

When automatically generating triggers from ERwin, an entity-level UDP (DeployData) indicates whether trigger code is generated for the static tables to support data deployment. A flag, called deploy_data, also exists in the entity_mnemonic table for the same purpose.

Table 3–5 lists the table’s FLA, fields, and foreign keys.

Table 3–5: gsc_deploy_dataset table information

Table FLA	Fields (data type)	Foreign keys
gscdd	deploy_dataset_obj (Decimal) dataset_code (Character) dataset_description (Character) disable_ri (Logical) source_code_data (Logical) deploy_full_data (Logical) xml_generation_procedure (Character) default_ado_filename (Character) deploy_additions_only (Logical) enable_data_versioning (Logical) deletion_dataset (Logical)	deploy_dataset_obj

Table 3–6 gives details of the table’s indexes.

Table 3–6: gsc_deploy_dataset index information

Index name	Elements	Type
XPKgsc_deploy_dataset	deploy_dataset_obj	Primary Unique
XAK1gsc_deploy_dataset	dataset_code	Unique
XIE1gsc_deploy_dataset	dataset_description	Nonunique
XIE2gsc_deploy_dataset	source_code_data	Nonunique

For customer sites that receive a dataset deployment, the last deployment loaded for this dataset is recorded on this table. This helps identify the version of the static data in a particular database.

A customer should not modify or deploy from the datasets sent by the application provider. Customers should create their own datasets containing the same tables and deploy from these datasets. A dataset deployment includes an XML file registered as part of the deployment. Customers can use that XML file for their databases and any other databases or sites to which they want to pass the data.

3.3.4 gsc_deploy_package table—gscdp

This table aids in defining groups of datasets that should be deployed together as a single package. This enables users to ensure they send all related datasets, including related or dependant data that has also changed.

Table 3–7 lists the table’s FLA, fields, and foreign keys.

Table 3–7: gsc_deploy_package table information

Table FLA	Fields (data type)	Foreign keys
gscdp	deploy_package_obj (Decimal) package_code (Character) package_description (Character)	deploy_package_obj

Table 3–8 gives details of the table’s indexes.

Table 3–8: gsc_deploy_package index information

Index name	Elements	Type
XPKgsc_deploy_package	deploy_package_obj	Primary Unique
XAK1gsc_deploy_package	package_code	Unique
XIE1gsc_deploy_package	package_description	Nonunique

3.3.5 gsc_package_dataset table—gscpd

This table defines the datasets that should be included with a package. A dataset might be included in multiple packages.

Table 3–9 lists the table’s FLA, fields, and foreign keys.

Table 3–9: gsc_package_dataset table information

Table FLA	Fields (data type)	Foreign keys
gscpd	package_dataset_obj (Decimal) deploy_dataset_obj (Decimal) deploy_package_obj (Decimal) deploy_full_data (Logical) package_dataset_seq (Integer)	deploy_dataset_obj deploy_package_obj

Table 3–10 gives details of the table’s indexes.

Table 3–10: gsc_package_dataset index information

Index name	Elements	Type
XPkgsc_package_dataset	package_dataset_obj	Primary Unique
XAK1gsc_package_dataset	deploy_dataset_obj deploy_package_obj	Unique
XIE2gsc_package_dataset	deploy_package_obj package_dataset_seq	Nonunique

3.3.6 gsc_scm_tool table—gscsm

This table defines possible software configuration management tools that could be integrated with Progress Dynamics. The tool currently in use is identified in the gsc_security_control table. An example of an integrated SCM tool is Roundtable (RTB). The primary purpose of this table is to link together crossreference information between the Repository and the SCM tool for data such as product modules and object types as specified in the gsm_scm_xref table.

Table 3–11 lists the table’s FLA, fields, and foreign keys.

Table 3–11: gsc_scm_tool table information

Table FLA	Fields (data type)	Foreign keys
gscsm	scm_tool_obj (Decimal) scm_tool_code (Character) scm_tool_description (Character)	scm_tool_obj

Table 3–12 gives details of the table’s indexes.

Table 3–12: gsc_scm_tool index information

Index name	Elements	Type
XPKgsc_scm_tool	scm_tool_obj	Primary Unique
XAK1gsc_scm_tool	scm_tool_code	Unique
XIE1gsc_scm_tool	scm_tool_description	Nonunique

3.3.7 gsm_release table—gsmrl

This table with its child table, gst_release_version, records what versions of an object belong to a release. This table identifies an actual version, with a specific reference, when the version was created, who created the version, and summary and detailed notes of the reason for the version.

The release number is usually generated automatically, using the gsc_sequence table to control the generation. The release number should contain the site number as part of the reference to avoid conflicts by indicating the site from which the release originated. When creating a release, a gst_release_version record must be created for every gst_record_version, marking the current version number of all data as of the release. This data can then be used to determine what data needs to be deployed between releases, the data that has been modified between the releases, by checking for matching version numbers in the gst_release_version table.

Table 3–13 lists the table’s FLA, fields, and foreign keys.

Table 3–13: gsm_release table information

Table FLA	Fields (data type)	Foreign keys
gsmrl	release_obj (Decimal) release_number (Character) release_summary (Character) release_notes (Character) release_date (Date) release_time (Integer) release_user (Character)	release_obj

Table 3–14 gives details of the table’s indexes.

Table 3–14: gsm_release index information

Index name	Elements	Type
XPkgsm_release	release_obj	Primary Unique
XAK1gsm_release	release_number	Unique
XIE1gsm_release	release_summary	Nonunique
XIE2gsm_release	release_notes	Nonunique
XIE3gsm_release	release_date release_time	Nonunique

3.3.8 gsm_scm_xref table—gsmsx

This table defines the mapping between data in the Repository and each external SCM tool being used. Examples of the data that is mapped are object types and product modules. This allows the external SCM tool to use different codes than those used in Progress Dynamics. It also allows multiple codes in Progress Dynamics to share a common code in the external SCM tool. For example, many product modules and object types in Progress Dynamics could point at common modules and subtypes in an SCM tool such as Roundtable.

The scm_foreign_key field is the field in the external SCM tool. It is a character field for maximum portability. You should use APIs to provide lookup lists for values in the external SCM tool. You must set up data in this table for SCM integration to function, so that it is clear what data is mapped to what. If it is a one-to-one mapping, then tools can be used to synchronize the data and set it up automatically.

Table 3–15 lists the table’s FLA, fields, and foreign keys.

Table 3–15: gsm_scm_xref table information

Table FLA	Fields (data type)	Foreign keys
gsmsx	scm_xref_obj (Decimal) scm_tool_obj (Decimal) owning_entity_mnemonic (Character) owning_reference (Character) owning_obj (Decimal) scm_foreign_key (Character)	scm_tool_obj

Table 3–16 gives details of the table’s indexes.

Table 3–16: gsm_scm_xref index information

Index name	Elements	Type
XPKgsm_scm_xref	scm_xref_obj	Primary Unique
XAK1gsm_scm_xref	scm_tool_obj owning_entity_mnemonic owning_reference	Unique
XIE1gsm_scm_xref	owning_entity_mnemonic	Nonunique
XIE2gsm_scm_xref	owning_obj	Nonunique
XIE3gsm_scm_xref	scm_foreign_key	Nonunique
XIE4gsm_scm_xref	owning_reference	Nonunique

3.3.9 **gst_dataset_file** table—gstdf

This table keeps a record of ADO files generated for a dataset. A single dataset can be generated out to multiple ADO files. This table records the date and time a dataset ADO file was last loaded into the current Repository. The date of the file on disk is compared to determine if new files have been downloaded that need to be updated into the local Repository.

If an ADO file is included as part of a package, this table records the package to which the ADO file belongs.

[Table 3–17](#) lists the table’s FLA, fields, and foreign keys.

Table 3–17: `gst_dataset_file` table information

Table FLA	Fields (data type)	Foreign keys
gstdf	dataset_file_obj (Decimal) deployment_obj (Decimal) deploy_dataset_obj (Decimal) ado_filename (Character) loaded_date (Date) loaded_time (Integer)	deploy_dataset_obj deployment_obj

[Table 3–18](#) gives details of the table’s indexes.

Table 3–18: `gst_dataset_file` index information

Index name	Elements	Type
XPkgst_dataset_file	dataset_file_obj	Primary Unique
XAK1gst_dataset_file	deployment_obj deploy_dataset_obj ado_filename	Unique
XIE2gst_dataset_file	deploy_dataset_obj	Nonunique
XIE3gst_dataset_file	ado_filename loaded_date loaded_time	Nonunique

3.3.10 **gst_record_version table—gstrv**

This table enables you to identify when static data is changed and needs to be deployed. This table is checked every time the deployment data is written to ensure that all data that matches the deployment criteria is written out.

When an item of data on a record changes, the replication trigger on the table checks if the version_data flag on the gsc_entity_mnemonic table is switched on.

If the flag is switched on, a record is written to this table. If a record already exists in the table, the record is updated to indicate that the data has changed by incrementing the version_number_seq and resetting the date, time, and user.

A version_number_seq greater than 0 indicates that the record has been changed locally and might need to be deployed. Once an import is done for a record, the version_number_seq is set back to 0, indicating the data has not been modified since the last import.

When importing data, the import_version_number_seq is used as a validity check. If this number does not match the import_version_number_seq of the data being imported, or the current version_number_seq does not match, there is a potential conflict.

Table 3–19 lists the table’s FLA, fields, and foreign keys.

Table 3–19: gst_record_version table information

Table FLA	Fields (data type)	Foreign keys
gstrv	record_version_obj (Decimal) entity_mnemonic (Character) key_field_value (Character) version_number_seq (Decimal) version_date (Date) version_time (Integer) version_user (Character) deletion_flag (Logical) import_version_number_seq (Decimal) last_version_number_seq (Decimal) secondary_key_value (Character)	entity_mnemonic record_version_obj

Table 3–20 gives details of the table’s indexes.

Table 3–20: gst_record_version index information

Index name	Elements	Type
XPkgst_record_version	record_version_obj	Primary Unique
XAK1gst_record_version	entity_mnemonic key_field_value	Unique
XIE1gst_record_version	key_field_value	Nonunique
XIE3gst_record_version	entity_mnemonic version_date version_time	Nonunique
XIE4gst_record_version	entity_mnemonic version_number_seq	Nonunique
XIE5gst_record_version	version_number_seq	Nonunique
XIE6gst_record_version	entity_mnemonic version_number_seq import_version_number_seq	Nonunique
XIE7gst_record_version	entity_mnemonic secondary_key_value	Nonunique

3.3.11 gst_release_version table—gstrl

This table records the record versions that make up a release. When creating a new release, a `gst_release_version` record is created for every `gst_record_version`. The `gst_release_version` record marks the current version number of all data as of the release. By checking for matching version numbers in the `gst_release_version` table, the framework can then determine what data needs to be deployed between releases, that is what data has been modified between the releases.

Table 3–21 lists the table’s FLA, fields, and foreign keys.

Table 3–21: gst_release_version table information

Table FLA	Fields (data type)	Foreign keys
gstrl	release_version_obj (Decimal) release_obj (Decimal) record_version_obj (Decimal) version_number_seq (Decimal) import_version_number_seq (Decimal) deletion_flag (Logical)	record_version_obj release_obj

Table 3–22 gives details of the table’s indexes.

Table 3–22: gst_release_version index information

Index name	Elements	Type
XPKgst_release_version	release_version_obj	Primary Unique
XAK1gst_release_version	release_obj record_version_obj	Unique

3.3.12 ryt_dbupdate_status table—rytds

This table audits what Progress Dynamics Conversion Utility (DCU) updates have occurred. It also controls the DCU update after completion of the DCU and allows it to be rerun or restarted as required.

To successfully run a conversion, the DCU must do several tasks. It must control and automate the tasks done for a specific release and the stage during which they should be run. It also needs to track whether each step completed successfully, the order of the steps, and other information.

Certain tasks can only occur after the DCU has finished. But they require a valid login to complete and must be finished before anybody starts using the system. Controlling this process is the main reason for this table. This table significantly improves the load-and-go functionality within Progress Dynamics and helps prevent migration and deployment issues.

The DCU uses the information in the .pf1 file to update this table.

Table 3–23 lists the table’s FLA, fields, and foreign keys.

Table 3–23: ryt_dbupdate_status table information

Table FLA	Fields (data type)	Foreign keys
rytds	dbupdate_status_obj (Decimal) update_db_name (Character) delta_version (Integer) file_type (Character) file_name (Character) update_when (Integer) run_sequence (Integer) update_completed (Logical) update_successful (Logical) return_result (Character) run_date (Date) run_time (Integer) run_by_user_obj (Decimal) update_new_db (Logical) update_existing_db (Logical) rerunnable (Logical) update_mandatory (Logical)	None

Table 3–24 gives details of the table’s indexes.

Table 3–24: ryt_dbupdate_status index information

Index name	Elements	Type
XPKryt_dbupdate_status	dbupdate_status_obj	Primary Unique
XAK1ryt_dbupdate_status	update_db_name delta_version file_type file_name update_when	Unique
XIE1ryt_dbupdate_status	update_completed update_when run_sequence	Nonunique
XIE2ryt_dbupdate_status	run_by_user_obj	Nonunique

Entity and Defaults Group

This chapter covers the group of tables in the Progress Dynamics Repository that store information about entities and defaults. This chapter covers the following topics:

- [Entity and Defaults group structure](#)
- [Table descriptions](#)

4.1 Entity and Defaults group structure

The Entity and Defaults group stores information about entity mnemonics and default settings for your application.

Figure 4–1 shows the structure and relationships of the tables in this group.

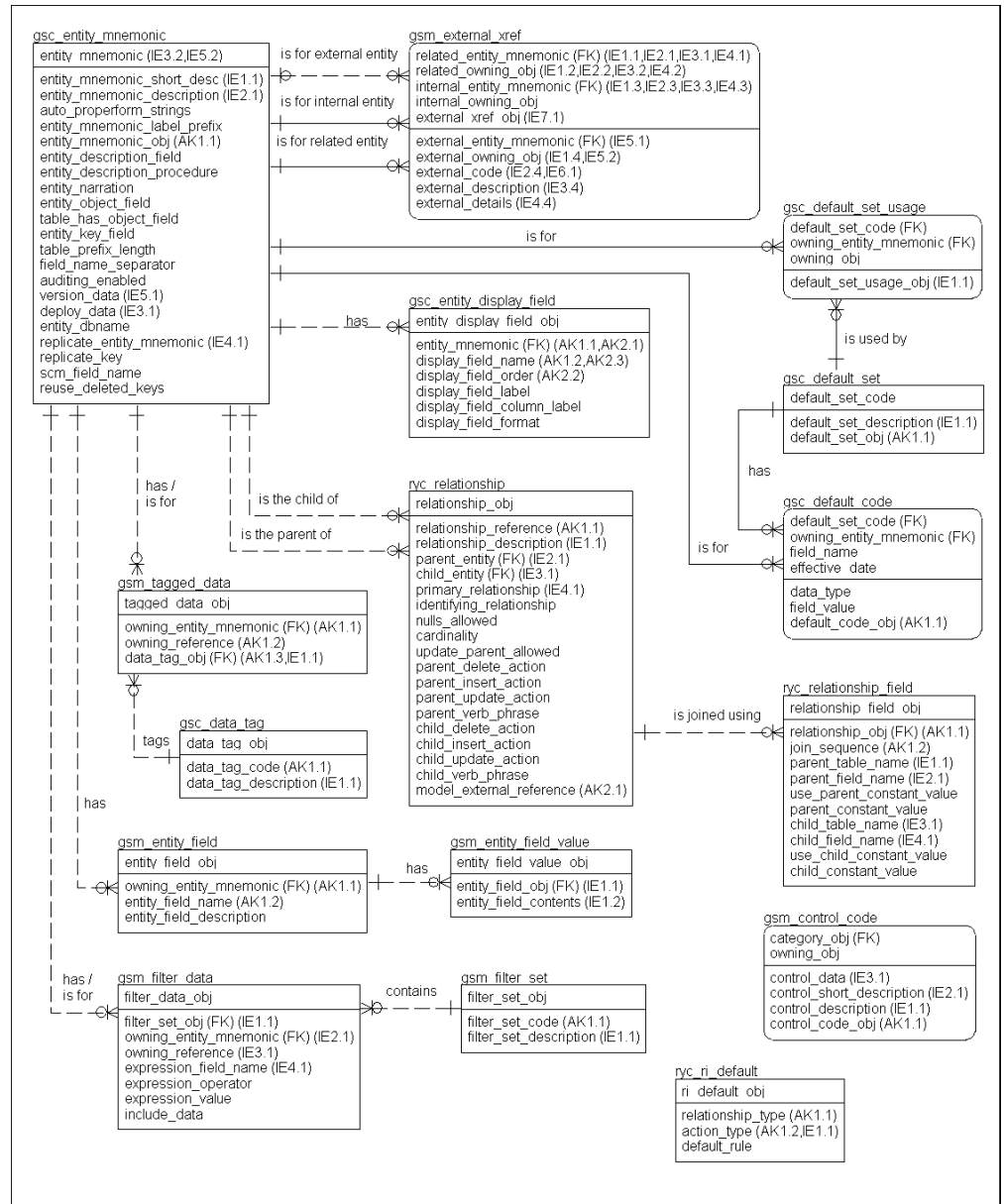


Figure 4–1: Entity and Defaults group

4.2 Table descriptions

The following sections provide detailed information about each table in the Entity and Defaults group. The Entity and Defaults group contains the following tables:

- [gsc_entity_mnemonic table—gscem](#)
- [gsc_data_tag table—gsctg](#)
- [gsc_default_code table—gscdc](#)
- [gsc_default_set table—gscds](#)
- [gsc_default_set_usage table—gscds](#)
- [gsc_entity_display_field table—gsced](#)
- [gsm_control_code table—gsmcl](#)
- [gsm_entity_field table—gsmef](#)
- [gsm_entity_field_value table—gsmev](#)
- [gsm_external_xref table—gsmex](#)
- [gsm_filter_data table—gsmfd](#)
- [gsm_filter_set table—gsmfi](#)
- [gsm_tagged_data table—gsmtf](#)
- [ryc_relationship table—rycre](#)
- [ryc_relationship_field table—rycrf](#)
- [ryc_ri_default table—rycri](#)

4.2.1 gsc_entity_mnemonic table—gschem

The gsc_entity_mnemonic table is the central table of the Entity and Defaults group. This table stores all the hard-coded entity mnemonics allocated to every table in an application. It defines a meaningful short code and identifies the table name for each table. It also defines generic information about the entity for use when generating dynamic objects or generic objects based on the table, and when auto-generating triggers.

Table 4–1 lists the table’s FLA, fields, and foreign keys.

Table 4–1: gsc_entity_mnemonic table information

Table FLA	Fields (data type)	Foreign keys
gschem	entity_mnemonic (Character) entity_mnemonic_short_desc (Character) entity_mnemonic_description (Character) auto_properform_strings (Logical) entity_mnemonic_label_prefix (Character) entity_mnemonic_obj (Character) entity_description_field (Character) entity_description_procedure (Character) entity_narration (Character) entity_object_field (Character) table_has_object_field (Logical) entity_key_field (Character) table_prefix_length (Integer) field_name_separator (Character) auditing_enabled (Logical) version_data (Logical) deploy_data (Logical) entity_dbname (Character) replicate_entity_mnemonic (Character) replicate_key (Character) scm_field_name (Character) reuse_deleted_keys (Logical)	entity_mnemonic

Table 4–2 gives details of the table’s indexes.

Table 4–2: gsc_entity_mnemonic index information

Index name	Elements	Type
XPKgsc_entity_mnemonic	entity_mnemonic	Primary Unique
XAK1gsc_entity_mnemonic	entity_mnemonic_obj	Unique
XIE1gsc_entity_mnemonic	entity_mnemonic_short_desc	Nonunique
XIE2gsc_entity_mnemonic	entity_mnemonic_description	Nonunique
XIE3gsc_entity_mnemonic	deploy_data entity_mnemonic	Nonunique
XIE4gsc_entity_mnemonic	replicate_entity_mnemonic	Nonunique
XIE5gsc_entity_mnemonic	version_data entity_mnemonic	Nonunique

4.2.2 gsc_data_tag table—gsctg

This table defines generic tags that can be assigned to data through the gsm_tagged_data table. For example, Progress Dynamics uses a tag "ry-own" to identify which data belongs to the framework and should not be modified by users of the framework. Applications can use the tag mechanism to tag data that belongs to specific applications, or for any other generic purpose.

The data in this table should be deployed as part of Progress Dynamics. It is typically only relevant at design and deployment time, rather than used as part of the run-time application. For example, you might apply rules to tagged data to prevent unauthorized or accidental modification of the data as design time. You might also use tags to help identify what data to deploy as part of an application.

Table 4–3 lists the table’s FLA, fields, and foreign keys.

Table 4–3: gsc_data_tag table information

Table FLA	Fields (data type)	Foreign keys
gsctg	data_tag_obj (Decimal) data_tag_code (Character) data_tag_description (Character)	data_tag_obj

Table 4–4 gives details of the table’s indexes.

Table 4–4: gsc_data_tag index information

Index name	Elements	Type
XPKgsc_data_tag	data_tag_obj	Primary Unique
XAKlgsc_data_tag	data_tag_code	Unique
XIElgsc_data_tag	data_tag_description	Nonunique

4.2.3 gsc_default_code table—gscdc

This table enables the creation of parameters or system defaults that need be specified to the system, without making changes to the database structure. These records can be grouped into sets using the gsc_default_set table, to create different parameter or defaults sets. Any use of defaults or sets of defaults must be hard-coded into an application. For example, you might have different parameter sets for warehouse control and administration groups.

Table 4–5 lists the table’s FLA, fields, and foreign keys.

Table 4–5: gsc_default_code table information

Table FLA	Fields (data type)	Foreign keys
gscdc	default_set_code (Character) owning_entity_mnemonic (Character) field_name (Character) effective_date (Date) data_type (Character) field_value (Character) default_code_obj (Decimal)	default_set_code field_name

[Table 4–6](#) gives details of the table’s indexes.

Table 4–6: gsc_default_code index information

Index name	Elements	Type
XPKgsc_default_code	default_set_code owning_entity_mnemonic field_name effective_date	Primary Unique
XAK1gsc_default_code	default_code_obj	Unique

4.2.4 gsc_default_set table—gscds

This table shows the associations between a set of parameters or defaults that are stored as gsc_default_code records. You could create a set of defaults that are applicable to the system in general, and other sets of defaults to be used in certain circumstances, such as for a particular department. You have to write code to make use of these sets in an application.

[Table 4–7](#) lists the table’s FLA, fields, and foreign keys.

Table 4–7: gsc_default_set table information

Table FLA	Fields (data type)	Foreign keys
gscds	default_set_code (Character) default_set_description (Character) default_set_obj (Decimal)	default_set_code

[Table 4–8](#) gives details of the table’s indexes.

Table 4–8: gsc_default_set index information

Index name	Elements	Type
XPKgsc_default_set	default_set_code	Primary Unique
XAK1gsc_default_set	default_set_obj	Unique
XIE1gsc_default_set	default_set_description	Nonunique

4.2.5 gsc_default_set_usage table—gscdu

This table associates default sets with objects in the application. For example, in a property administration application, you could have a default set for each administration company.

Table 4–9 lists the table’s FLA, fields, and foreign keys.

Table 4–9: gsc_default_set_usage table information

Table FLA	Fields (data type)	Foreign keys
gscdu	default_set_code (Character) owning_entity_mnemonic (Character) owning_obj (Decimal) default_set_usage_obj (Decimal)	default_set_code

Table 4–10 gives details of the table’s indexes.

Table 4–10: gsc_default_set_usage index information

Index name	Elements	Type
XPKgsc_default_set_usage	default_set_code owning_entity_mnemonic owning_obj	Primary Unique
XIE1gsc_default_set_usage	default_set_usage_obj	Nonunique

4.2.6 gsc_entity_display_field table—gsced

This table defines the fields in a table to be used when building generic objects for that table, such as a dynamic browser of the table. It identifies the fields that should be used, their sequence, and enables overriding the fields’ labels and formats.

This is used in the generic data security used by the Progress Dynamics framework.

If there are no entries in this table, the default behavior is to use all fields, other than object fields, in the order they appear in the database.

This table should initially be populated automatically from the metaschema. It can then be modified accordingly.

This table does not support joined fields.

Table 4–11 lists the table’s FLA, fields, and foreign keys.

Table 4–11: gsc_entity_display_field table information

Table FLA	Fields (data type)	Foreign keys
gsced	entity_display_field_obj (Decimal) entity_mnemonic (Character) display_field_name (Character) display_field_order (Integer) display_field_label (Character) display_field_column_label (Character) display_field_format (Character)	entity_mnemonic

Table 4–12 gives details of the table’s indexes.

Table 4–12: gsc_entity_display_field index information

Index name	Elements	Type
XPKgsc_entity_display_field	entity_display_field_obj	Primary Unique
XAK1gsc_entity_display_field	entity_mnemonic display_field_name	Unique
XAK2gsc_entity_display_field	entity_mnemonic display_field_order display_field_name	Unique

4.2.7 **gsm_control_code table—gsmcl**

This is a generic table for holding device control codes. The category is used to define the purpose of the code. In addition, the owning_obj can optionally be used to define the device to which the code relates. If the owning_obj is left as 0, then it applies to all devices.

Table 4–13 lists the table’s FLA, fields, and foreign keys.

Table 4–13: gsm_control_code table information

Table FLA	Fields (data type)	Foreign keys
gsmcl	category_obj (Decimal) owning_obj (Decimal) control_data (Character) control_short_description (Character) control_description (Character) control_code_obj (Decimal)	category_obj owning_obj

Table 4–14 gives details of the table’s indexes.

Table 4–14: gsm_control_code index information

Index name	Elements	Type
XPKgsm_control_code	category_obj owning_obj	Primary Unique
XAK1gsm_control_code	control_code_obj	Unique
XIE1gsm_control_code	control_description	Nonunique
XIE2gsm_control_code	control_short_description	Nonunique
XIE3gsm_control_code	control_data	Nonunique

This might be used in a point-of-sale system where codes must be sent to a pole for various reasons, such as resetting the poll or making the message scroll. Different categories can be defined for each action. The owning_entity_mnemonic on the category determines the table to which the owning_obj relates. This is usually an application-specific device table.

4.2.8 **gsm_entity_field table—gsmeff**

This table aids in securing any application-specific data or generic data against any entity.

Table 4–15 lists the table’s FLA, fields, and foreign keys.

Table 4–15: gsm_entity_field table information

Table FLA	Fields (data type)	Foreign keys
gsmeff	entity_field_obj (Decimal) owning_entity_mnemonic (Character) entity_field_name (Character) entity_field_description (Character)	entity_field_obj

Table 4–16 gives details of the table’s indexes.

Table 4–16: gsm_entity_field index information

Index name	Elements	Type
XPKgsm_entity_field	entity_field_obj	Primary Unique
XAK1gsm_entity_field	owning_entity_mnemonic entity_field_name	Unique

For example, if there is a requirement to secure access to specific companies, then the company entity with the company code field could be set up in this table. The valid values could then be set up in the entity field values table, and users allocated access to the specific values.

4.2.9 **gsm_entity_field_value table—gsmevf**

This table lists the valid values for the entity field, such as company codes.

Table 4–17 lists the table’s FLA, fields, and foreign keys.

Table 4–17: gsm_entity_field_value table information

Table FLA	Fields (data type)	Foreign keys
gsmevf	entity_field_value_obj (Decimal) entity_field_obj (Decimal) entity_field_contents (Character)	entity_field_obj

Table 4–18 gives details of the table’s indexes.

Table 4–18: gsm_entity_field_value index information

Index name	Elements	Type
XPkgsm_entity_field_value	entity_field_value_obj	Primary Unique
XIE1gsm_entity_field_value	entity_field_obj entity_field_contents	Unique

4.2.10 gsm_external_xref table—gsmex

This table defines generic cross-reference information to details in external tables.

For example, an organization whose accounts are held in the database might have other account codes in an external database from which the account codes originated. This table could be used to define which internal accounts point to which external accounts for cross-referencing and reporting purposes. For this example, the fields are set up as follows:

- The related entity is the gsm_organization table in the database.
- The related object is a specific organization.
- The internal entity is the gsm_account table in the database.
- The internal object is a specific account code.

Table 4–19 lists the table’s FLA, fields, and foreign keys.

Table 4–19: gsm_external_xref table information

Table FLA	Fields (data type)	Foreign keys
gsmex	related_entity_mnemonic (Character) related_owning_obj (Decimal) internal_entity_mnemonic (Character) internal_owning_obj (Decimal) external_xref_obj (Decimal) external_entity_mnemonic (Character) external_owning_obj (Decimal) external_code (Character) external_description (Character) external_details (Character)	None

Table 4–20 gives details of the table’s indexes.

Table 4–20: gsm_external_xref index information

Index name	Elements	Type
XPKgsm_external_xref	related_entity_mnemonic related_owning_obj internal_entity_mnemonic internal_owning_obj external_xref_obj	Primary Unique
XIE1gsm_external_xref	related_entity_mnemonic related_owning_obj internal_entity_mnemonic external_owning_obj	Nonunique
XIE2gsm_external_xref	related_entity_mnemonic related_owning_obj internal_entity_mnemonic external_code	Nonunique
XIE3gsm_external_xref	related_entity_mnemonic related_owning_obj internal_entity_mnemonic external_description	Nonunique
XIE4gsm_external_xref	related_entity_mnemonic related_owning_obj internal_entity_mnemonic external_details	Nonunique
XIE5gsm_external_xref	external_entity_mnemonic external_owning_obj	Nonunique
XIE6gsm_external_xref	external_code	Nonunique
XIE7gsm_external_xref	external_xref_obj	Nonunique

If an external table is available, then the external entity and object can be defined, otherwise the external details are keyed directly into this table.

4.2.11 **gsm_filter_data table—gsmfd**

This table defines the filters that apply to a filter set. The table is meant to exclude all specified data except the data that is specifically included back in. For example, exclude all objects in all Repository modules, except for objects where the template flag is true or the object type is a toolbar.

By default, the include_data flag is set to NO specifying that criteria must be excluded from the result set. If the include_data flag is set to YES, it is treated as an override condition to re-include specific data. This works by building up a bracketed WHERE clause for all the exclusions using the AND operator. It then adds an OR operator for any data that must be re-included outside the bracket to override for specific data. Use the include_data flag sparingly to avoid performance problems.

To specify a certain record, the owning_reference field points at the object ID for that table. For example, if the owning_entity_mnemonic is GSCPM for gsc_product_module, then the owning_reference points to a specific product module, product_module_obj. You can also specify a more generic expression by supplying a field name such as object_filename, an operator such as BEGINS, and a value such as “standardtoolbar.” You cannot specify a generic expression and a specific record, one or the other must be specified in a single filter data record.

This table provides significant flexibility in the specification of what data to filter. For example, it is possible to specify a list of product modules to exclude, but to re-include specific objects from some of the excluded product modules.

[Table 4–21](#) lists the table’s FLA, fields, and foreign keys.

Table 4–21: gsm_filter_data table information

Table FLA	Fields (data type)	Foreign keys
gsmfd	filter_data_obj (Decimal) filter_set_obj (Decimal) owning_entity_mnemonic (Character) owning_reference (Character) expression_field_name (Character) expression_operator (Character) expression_value (Character) include_data (Logical)	filter_set_obj

Table 4–22 gives details of the table’s indexes.

Table 4–22: **gsm_filter_data** index information

Index name	Elements	Type
XPKgsm_filter_data	filter_data_obj	Primary Unique
XIE1gsm_filter_data	filter_set_obj	Nonunique
XIE2gsm_filter_data	owning_entity_mnemonic	Nonunique
XIE3gsm_filter_data	owning_reference	Nonunique
XIE4gsm_filter_data	expression_field_name	Nonunique

4.2.12 gsm_filter_set table—gsmfi

This table groups the filter settings that make up a single filter definition. For example, a standard filter set called “Repository” contains all the filter records required to filter out repository data. Filter sets can be modified and extended to filter out additional data or include data that was previously excluded. User profile codes are used to assign a filter set to a user, rather than setting the flag to display repository data. If no filter set is allocated to a user, then no filters apply. Only a single filter set can be applied at any time.

Table 4–23 lists the table’s FLA, fields, and foreign keys.

Table 4–23: **gsm_filter_set** table information

Table FLA	Fields (data type)	Foreign keys
gsmfi	filter_set_obj (Decimal) filter_set_code (Character) filter_set_description (Character)	filter_set_obj

Table 4–24 gives details of the table’s indexes.

Table 4–24: gsm_filter_set index information

Index name	Elements	Type
XPKgsm_filter_set	filter_set_obj	Primary Unique
XAKlgsm_filter_set	filter_set_code	Unique
XIElgsm_filter_set	filter_set_description	Nonunique

4.2.13 gsm_tagged_data table—gsmtdd

This table associates data tags defined in the gsc_data_tag table with specific items of data, such as, specific object types to identify that the object types belong to the framework itself. A specific item of data may contain any number of tags for various purposes, but a specific tag can only be allocated once to a specific item of data.

As tags are generically attached to data, the schema triggers for data that might have tags attached needs to ensure that the tags are deleted when the data is deleted.

Table 4–25 lists the table’s FLA, fields, and foreign keys.

Table 4–25: gsm_tagged_data table information

Table FLA	Fields (data type)	Foreign keys
gsmtdd	tagged_data_obj (Decimal) owning_entity_mnemonic (Character) owning_reference (Character) data_tag_obj (Decimal)	data_tag_obj

Table 4–26 gives details of the table’s indexes.

Table 4–26: **gsm_tagged_data** index information

Index name	Elements	Type
XPKgsm_tagged_data	tagged_data_obj	Primary Unique
XAK1gsm_tagged_data	owning_reference owning_entity_mnemonic data_tag_obj	Unique
XIE1gsm_tagged_data	data_tag_obj	Nonunique

4.2.14 ryc_relationship table—rycre

This table stores relationship information for tables in the Repository and application databases built using Progress Dynamics.

Table 4–27 lists the table’s FLA, fields, and foreign keys.

Table 4–27: **ryc_relationship** table information

Table FLA	Fields (data type)	Foreign keys
rycre	relationship_obj (Decimal) relationship_reference (Character) relationship_description (Character) parent_entity (Character) child_entity (Character) primary_relationship (Logical) identifying_relationship (Logical) nulls_allowed (Logical) cardinality (Character) update_parent_allowed (Logical) parent_delete_action (Character) parent_insert_action (Character) parent_update_action (Character) parent_verb_phrase (Character) child_delete_action (Character) child_insert_action (Character) child_update_action (Character) child_verb_phrase (Character) model_external_reference (Character)	relationship_obj

Table 4–28 gives details of the table’s indexes.

Table 4–28: ryc_relationship index information

Index name	Elements	Type
XPKryc_relationship	relationship_obj	Primary Unique
XAK1ryc_relationship	relationship_reference	Unique
XAK2ryc_relationship	model_external_reference	Unique
XIE1ryc_relationship	relationship_description	Nonunique
XIE2ryc_relationship	parent_entity	Nonunique
XIE3ryc_relationship	child_entity	Nonunique
XIE4ryc_relationship	primary_relationship	Nonunique

Multiple relationships can exist between the same parent and child table if required, so one of these must be flagged as the primary relationship to use initially when joining between the two tables.

The relationship reference field is unique so that it can be referenced in code if required, where multiple possible relationships exists and application functionality depends on the relationship. Where the relationship reference is irrelevant, it can be automatically generated using the Progress Dynamics sequences.

The attributes about the relationship map closely with the attributes supported by *ERwin* from Computer Associates.

The fields used to join the tables in the relationship are specified in the child table `ryc_relationship_field`.

The contents of this table should be automatically populated from information exported from a case tool, such as *ERwin*, to make synchronization of changes as automated as possible.

This table, once populated, can be used to support generic application functionality such as automatic object generation, referential integrity trigger code, etc.

4.2.15 ryc_relationship_field table—rycrf

This table defines the fields used to join the two tables defined in the ryc_relationship table that this table is a child of. Multiple field joins are supported, as well as rolenamed foreign keys where the field names in the two tables do not match.

Table 4–29 lists the table’s FLA, fields, and foreign keys.

Table 4–29: ryc_relationship_field table information

Table FLA	Fields (data type)	Foreign keys
rycrf	relationship_field_obj (Decimal) relationship_obj (Decimal) join_sequence (Integer) parent_table_name (Character) parent_field_name (Character) use_parent_constant_value (Logical) parent_constant_value (Character) child_table_name (Character) child_field_name (Character) use_child_constant_value (Logical) child_constant_value (Character)	relationship_obj

Table 4–30 gives details of the table’s indexes.

Table 4–30: ryc_relationship_field index information

Index name	Elements	Type
XPKryc_relationship_field	relationship_field_obj	Primary Unique
XAK1ryc_relationship_field	relationship_obj join_sequence	Unique
XIE1ryc_relationship_field	parent_table_name	Nonunique
XIE2ryc_relationship_field	parent_field_name	Nonunique
XIE3ryc_relationship_field	child_table_name	Nonunique
XIE4ryc_relationship_field	child_field_name	Nonunique

The join sequence determines the order to reference the fields when constructing a dynamic where clause to join the tables.

When joining to some tables, additional constant values for fields in the child table or parent table might need to be specified. This functionality is supported.

For example, consider a join from the Repository's `gsc_object_type` table to the `ryc_attribute_value` table. There is a single field from the parent table, the `object_type_obj`. But you must additionally specify a 0 value for other fields, such as, `container_smartobject_obj`, `smartobject_obj`, and `object_instance_obj`. This means that when using constant values, the child or the parent field can be left blank.

4.2.16 `ryc_ri_default` table—`rycri`

This table defines the default referential integrity (RI) rules to apply when manually maintaining relationships in the `ryc_relationship` table.

[Table 4–31](#) lists the table's FLA, fields, and foreign keys.

Table 4–31: `ryc_ri_default` table information

Table FLA	Fields (data type)	Foreign keys
rycri	ri_default_obj (Decimal) relationship_type (Character) action_type (Character) default_rule (Character)	None

[Table 4–32](#) gives details of the table's indexes.

Table 4–32: `ryc_ri_default` index information

Index name	Elements	Type
XPKryc_ri_default	ri_default_obj	Primary Unique
XAK1ryc_ri_default	relationship_type action_type	Unique
XIE1ryc_ri_default	action_type	Nonunique

The rules in this table are used to default the parent and child actions based on the value of the `identifying_relationship` and `nulls_allowed` fields.

The standard set of referential integrity defaults used by Progress Dynamics are as follows:

- For identifying and nonidentifying but with no nulls allowed the defaults are:
 - Child Delete = None
 - Child Insert = Restrict
 - Child Update = Restrict
 - Parent Delete = Restrict
 - Parent Insert = None
 - Parent Update = Restrict
- For nonidentifying but with nulls allowed the defaults are:
 - Child Delete = None
 - Child Insert = Set Null
 - Child Update = Set Null
 - Parent Delete = Set Null
 - Parent Insert = None
 - Parent Update = Set Null

These are usually correct apart from the need to sometimes change the delete rules for the parent to cascade. Sometimes even when a relationship allows nulls, you still want a restrict rule rather than a set null rule.

Error Group

This chapter covers the group of tables in the Progress Dynamics Repository that store information about error handling in your application. This chapter covers the following topics:

- [Error group structure](#)
- [Table descriptions](#)

5.1 Error group structure

The Error group stores information about error handling in your application. [Figure 5–1](#) shows the structure and relationships of the tables in this group.

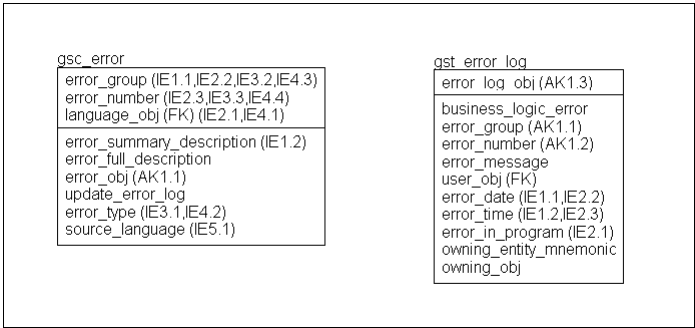


Figure 5–1: Error group

5.2 Table descriptions

The following sections provide detailed information about each table in the Error group. The Error group contains the following tables:

- [gsc_error table—gscer](#)
- [gst_error_log table—gster](#)

5.2.1 gsc_error table—gscer

This table defines all the application errors that can occur, providing summary and full descriptions for the errors. The summary description is shown first, with an option to display a longer description, if available. The use of error codes from this table aids in customizing error messages.

Table 5–1 lists the table’s FLA, fields, and foreign keys.

Table 5–1: gsc_error table information

Table FLA	Fields (data type)	Foreign keys
gscer	error_group (Character) error_number (Integer) language_obj (Decimal) error_summary_description (Character) error_full_description (Character) error_obj (Decimal) update_error_log (Logical) error_type(Character) source_language (Logical)	language_obj

Table 5–2 gives details of the table’s indexes.

Table 5–2: gsc_error index information

Index name	Elements	Type
XPkgsc_error	error_group error_number language_obj	Primary Unique
XAK1gsc_error	error_obj	Unique
XIE1gsc_error	error_group error_summary_description	Nonunique
XIE2gsc_error	language_obj error_group error_number	Nonunique
XIE3gsc_error	error_type error_group error_number	Nonunique
XIE4gsc_error	language_obj error_type error_group error_number	Nonunique
XIE5gsc_error	source_language	Nonunique

This table supports any kind of message to the user. Messages should not be hard-coded in the application. Every message to the user should use this mechanism. Supported message types are as follows:

- **MES**—Message
- **INF**—Information
- **ERR**—Error
- **WAR**—Warning
- **QUE**—Question

The default is ERR for Error if nothing is set up.

Errors in multiple languages are supported, if required.

5.2.2 gst_error_log table—gster

This table holds a list of errors generated either from business logic or user interface code.

[Table 5–3](#) lists the table’s FLA, fields, and foreign keys.

Table 5–3: gst_error_log table information

Table FLA	Fields (data type)	Foreign keys
gster	error_log_obj (Decimal) business_logic_error (Logical) error_group (Character) error_number (Integer) error_message (Character) user_obj (Decimal) error_date (Date) error_time (Integer) error_in_program (Character) owning_entity_mnemonic (Character) owning_obj (Decimal)	user_obj

Table 5–4 gives details of the table’s indexes.

Table 5–4: gst_error_log index information

Index name	Elements	Type
XPKgst_error_log	error_log_obj	Primary Unique
XAK1gst_error_log	error_group error_number error_log_obj	Unique
XIE1gst_error_log	error_date error_time	Nonunique
XIE2gst_error_log	error_in_program error_date error_time	Nonunique

The table is periodically archived to ensure it does not get too large.

Because writing directly to this table would form part of a transaction being undone, data has to be gathered indirectly. The data in the table is gathered directly from the user interface. Data is also gathered periodically from the business logic error file, which is a flat file.

Globalization Group

This chapter covers the group of tables in the Progress Dynamics Repository that store information used in deploying your application internationally. This chapter covers the following topics:

- [Globalization group structure](#)
- [Managers](#)
- [Table descriptions](#)

6.1 Globalization group structure

The Globalization group stores information used to globalize your application. [Figure 6–1](#) shows the structure and relationships of the tables in this group.

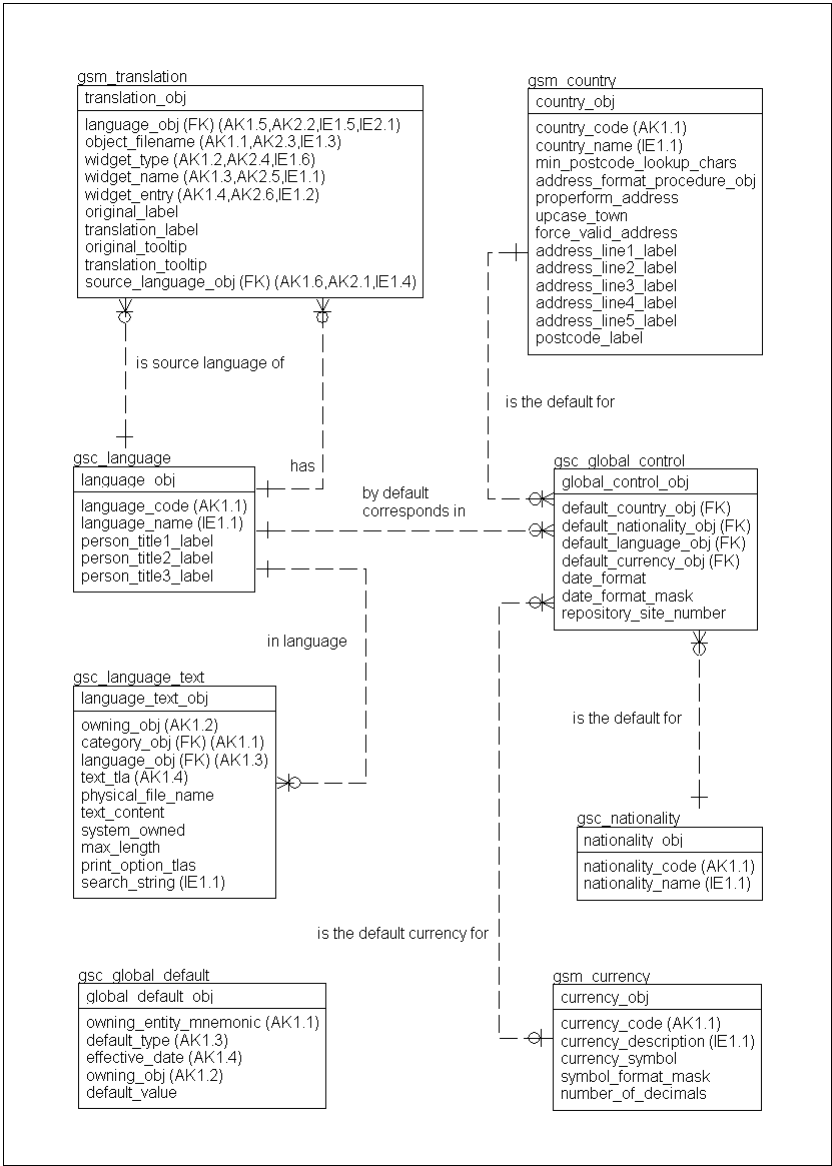


Figure 6–1: Globalization group

6.2 Managers

The Localization Manager uses data from these tables to perform run-time translations of objects in the context of an application screen. For more detail about the temp-tables and APIs the Localization Manager uses to manipulate data from the Globalization group tables, see the chapter on the Localization Manager in the *Progress Dynamics Managers API Reference*.

6.3 Table descriptions

The following sections provide detailed information about each table in the Globalization group. The Globalization group contains the following tables:

- [gsc_language table—gsclg](#)
- [gsc_global_control table—gscgc](#)
- [gsc_global_default table—gscgd](#)
- [gsc_language_text table—gsclt](#)
- [gsc_nationality table—gscna](#)
- [gsm_country table—gsmcy](#)
- [gsm_currency table—gsmcr](#)
- [gsm_translation table—gsmtl](#)

6.3.1 gsc_language table—gsc1g

The gsc_language table is the central table of the Globalization group. This table lists the languages supported by the system.

[Table 6–1](#) lists the table’s FLA, fields, and foreign keys.

Table 6–1: gsc_language table information

Table FLA	Fields (data type)	Foreign keys
gsc1g	language_obj (Decimal) language_code (Character) language_name (Character) person_title1_label (Character) person_title2_label (Character) person_title3_label (Character)	language_obj

[Table 6–2](#) gives details of the table’s indexes.

Table 6–2: gsc_language index information

Index name	Elements	Type
XPKgsc_language	language_obj	Primary Unique
XAK1gsc_language	language_code	Unique
XIE1gsc_language	language_name	Nonunique

6.3.2 gsc_global_control table—gscgc

This table defines system-wide defaults. It contains a single record, which holds the current system defaults.

[Table 6–3](#) lists the table’s FLA, fields, and foreign keys.

Table 6–3: gsc_global_control table information

Table FLA	Fields (data type)	Foreign keys
gscgc	global_control_obj (Decimal) default_country_obj (Decimal) default_nationality_obj (Decimal) default_language_obj (Decimal) default_currency_obj (Decimal) date_format (Character) date_format_mask (Character) repository_site_number (Integer)	None

[Table 6–4](#) gives details of the table’s indexes.

Table 6–4: gsc_global_control index information

Index name	Elements	Type
XPkgsc_global_control	global_control_obj	Primary Unique

6.3.3 gsc_global_default table—gscgd

This table contains global default values across the entire application. The parameter file used to start the application is specific to a given user. The entries in this table are system wide.

Table 6–5 lists the table’s FLA, fields, and foreign keys.

Table 6–5: gsc_global_default table information

Table FLA	Fields (data type)	Foreign keys
gscgd	global_default_obj (Decimal) owning_entity_mnemonic (Character) default_type (Character) effective_date (Date) owning_obj (Decimal) default_value (Character)	None

Table 6–6 gives details of the table’s indexes.

Table 6–6: gsc_global_default index information

Index name	Elements	Type
XPKgsc_global_default	global_default_obj	Primary Unique
XAK1gsc_global_default	owning_entity_mnemonic owning_obj default_type effective_date	Unique

Standard entries exist in the gsc_global_control table. This table aids in adding other generic defaults without the need for database changes. The entries in this table are system-owned by their nature. The only fields that can change are owning_obj and default_value. Changing any of these values creates a new record for the owning_entity_mnemonic and default_type, effective as of the new date with the new values.

Only a system administrator can delete entries from this table.

6.3.4 gsc_language_text table—gsclt

This table contains generic text files for all supported languages. Texts might be associated with another entity by the owning_obj field, or might be generic text of a certain type. Numbers enclosed in brackets, { }, are used for parameter substitutions, such as Scheme option names, transaction narrations, and valid people titles.

Table 6–7 lists the table’s FLA, fields, and foreign keys.

Table 6–7: gsc_language_text table information

Table FLA	Fields (data type)	Foreign keys
gsclt	language_text_obj (Decimal) owning_obj (Decimal) category_obj (Decimal) language_obj (Decimal) text_tla (Character) physical_file_name (Character) text_content (Character) system_owned (Logical) max_length (Integer) print_option_tlas (Character) search_string (Character)	category_obj language_obj owning_obj

Table 6–8 gives details of the table’s indexes.

Table 6–8: gsc_language_text index information

Index name	Elements	Type
XPKgsc_language_text	language_text_obj	Primary Unique
XAKlgsc_language_text	category_obj owning_obj language_obj text_tla	Unique
XIElgsc_language_text	search_string	Nonunique

6.3.5 gsc_nationality table—gscna

This table lists the allowable nationalities.

[Table 6–9](#) lists the table’s FLA, fields, and foreign keys.

Table 6–9: gsc_nationality table information

Table FLA	Fields (data type)	Foreign keys
gscna	nationality_obj (Decimal) nationality_code (Character) nationality_name (Character)	None

[Table 6–10](#) gives details of the table’s indexes.

Table 6–10: gsc_nationality index information

Index name	Elements	Type
XPKgsc_nationality	nationality_obj	Primary Unique
XAK1gsc_nationality	nationality_code	Unique
XIE1gsc_nationality	nationality_name	Nonunique

6.3.6 **gsm_country table—gsmcy**

This table lists the supported countries, for example, USA = United States of America, SA = South Africa, and UK = United Kingdom.

[Table 6–11](#) lists the table’s FLA, fields, and foreign keys.

Table 6–11: gsm_country table information

Table FLA	Fields (data type)	Foreign keys
gsmcy	country_obj (Decimal) country_code (Character) country_name (Character) min_postcode_lookup_chars (Integer) address_format_procedure_obj (Decimal) properform_address (Logical) upcase_town (Logical) force_valid_address (Logical) address_line1_label (Character) address_line2_label (Character) address_line3_label (Character) address_line4_label (Character) address_line5_label (Character) postcode_label (Character)	None

[Table 6–12](#) gives details of the table’s indexes.

Table 6–12: gsm_country index information

Index name	Elements	Type
XPkgsm_country	country_obj	Primary Unique
XAK1gsm_country	country_code	Unique
XIE1gsm_country	country_name	Nonunique

6.3.7 `gsm_currency` table—`gsmcr`

This table contains all the currency codes and their symbol references that are available to the system.

[Table 6–13](#) lists the table’s FLA, fields, and foreign keys.

Table 6–13: `gsm_currency` table information

Table FLA	Fields (data type)	Foreign keys
<code>gsmcr</code>	<code>currency_obj</code> (Decimal) <code>currency_code</code> (Character) <code>currency_description</code> (Character) <code>currency_symbol</code> (Character) <code>symbol_format_mask</code> (Character) <code>number_of_decimals</code> (Integer)	None

[Table 6–14](#) gives details of the table’s indexes.

Table 6–14: `gsm_currency` index information

Index name	Elements	Type
<code>XPKgsm_currency</code>	<code>currency_obj</code>	Primary Unique
<code>XAK1gsm_currency</code>	<code>currency_code</code>	Unique
<code>XIE1gsm_currency</code>	<code>currency_description</code>	Nonunique

6.3.8 **gsm_translation table—gsmtl**

This table contains user-defined translations in various languages for widget labels and ToolTip text. The setup of every program walks the widget tree. If an entry exists in this table, the label and ToolTip are changed to the entry in this table according to the language selected by the user.

Translations can be turned off globally using the `translation_enabled` field on the `gsc_security_control` table.

[Table 6–15](#) lists the table’s FLA, fields, and foreign keys.

Table 6–15: gsm_translation table information

Table FLA	Fields (data type)	Foreign keys
gsmtl	translation_obj (Decimal) language_obj (Decimal) object_filename (Character) widget_type (Character) widget_name (Character) widget_entry (Integer) original_label (Character) translation_label (Character) original_tooltip (Character) translation_tooltip (Character) source_language_obj (Decimal)	language_obj

Table 6–16 gives details of the table’s indexes.

Table 6–16: gsm_translation index information

Index name	Elements	Type
XPKgsm_translation	translation_obj	Primary Unique
XAK1gsm_translation	object_filename widget_type widget_name widget_entry language_obj source_language_obj	Unique
XAK2gsm_translation	source_language_obj language_obj object_filename widget_type widget_name widget_entry	Unique
XIE1gsm_translation	widget_name widget_entry object_filename source_language_obj language_obj widget_type	Nonunique
XIE2gsm_translation	language_obj	Nonunique

Menu and Toolbar Group

This chapter covers the group of tables in the Progress Dynamics Repository that store information about menus and toolbars. This chapter covers the following topics:

- [Menu and Toolbar group structure](#)
- [Managers](#)
- [Table descriptions](#)

7.1 Menu and Toolbar group structure

The Menu and Toolbar group stores information about the menus and toolbars in your application.

Figure 7–1 shows the structure and relationships of the tables in this group.

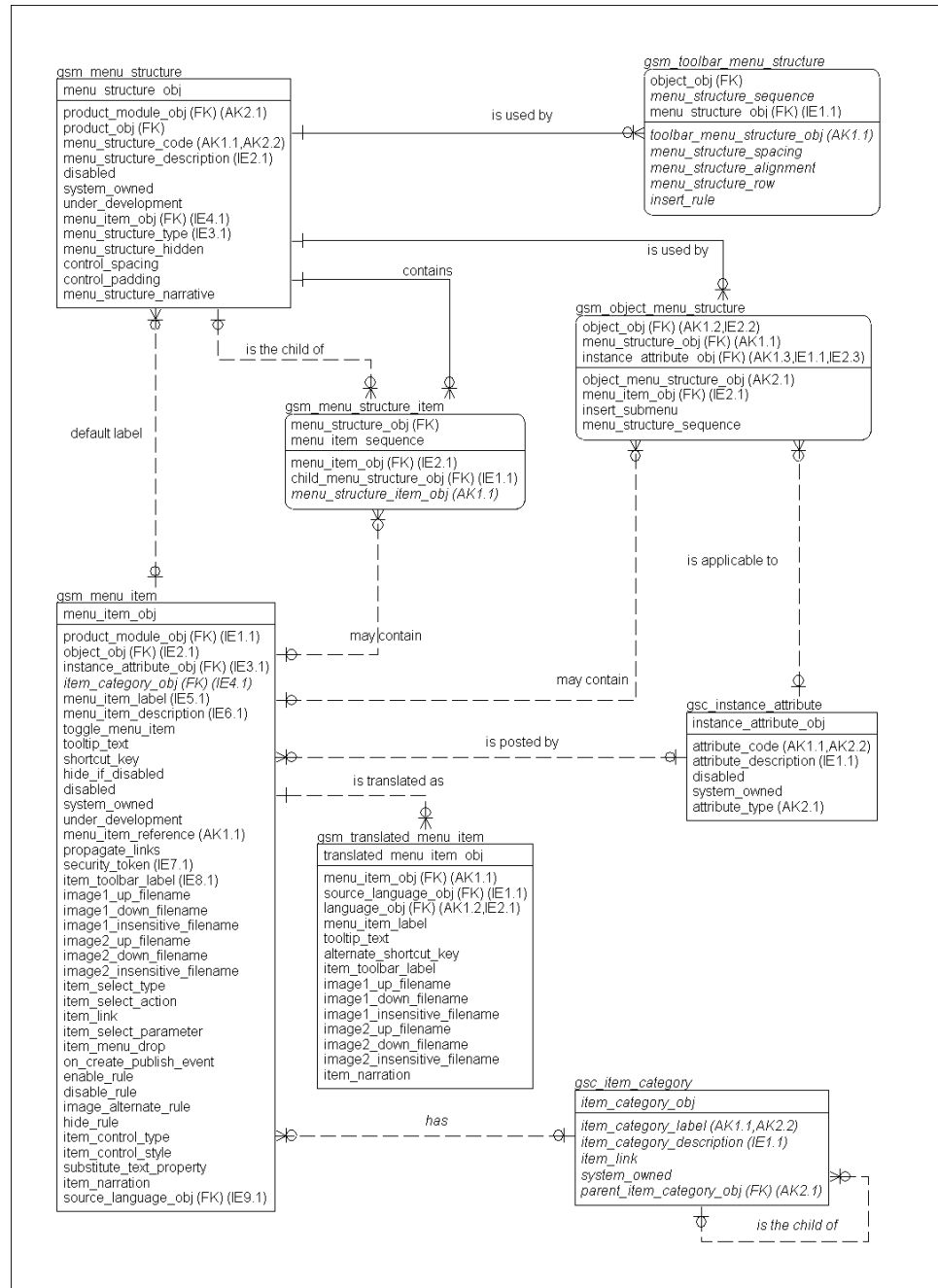


Figure 7–1: Menu and Toolbar group

7.2 Managers

The Repository Managers use these tables to provide the Progress Dynamics framework with information to build application user interfaces. For more detail about the temp-tables and APIs the Repository Managers use to manipulate data from the Menu and Toolbar group tables, see the chapter on the Repository Managers in the [Progress Dynamics Managers API Reference](#).

7.3 Table descriptions

The following sections provide detailed information about each table in the Menu and Toolbar group. The Menu and Toolbar group contains the following tables:

- [gsm_menu_item table—gsmmi](#)
- [gsc_instance_attribute table—gscia](#)
- [gsc_item_category table—gscic](#)
- [gsm_menu_structure table—gsmps](#)
- [gsm_menu_structure_item table—gsmit](#)
- [gsm_object_menu_structure table—gsmom](#)
- [gsm_toolbar_menu_structure table—gsmtm](#)
- [gsm_translated_menu_item table—gsmti](#)

7.3.1 **gsm_menu_item table—gsmmi**

The `gsm_menu_item` table is the central table of the Menu and Toolbar group.

This table defines the dynamic menu items that might belong to either a menu structure or a toolbar. On a menubar, an “item” can be visualized as a menu item, a submenu, or a ruler. On a toolbar, an “item” can be visualized as a control, a button, or a combo box. A menu item might launch an actual program, publish an event to an object, or set a property.

Table 7–1 lists the table’s FLA, fields, and foreign keys.

Table 7–1: `gsm_menu_item` table information

Table FLA	Fields (data type)	Foreign keys
gsmmi	menu_item_obj (Decimal) product_module_obj (Decimal) object_obj (Decimal) instance_attribute_obj (Decimal) item_category_obj (Decimal) menu_item_label (Character) menu_item_description (Character) toggle_menu_item (Logical) tooltip_text (Character) shortcut_key (Character) hide_if_disabled (Logical) disabled (Logical) system_owned (Logical) under_development (Logical) menu_item_reference (Character) propagate_links (Character) security_token (Character) item_toolbar_label (Character) image1_up_filename (Character) image1_down_filename (Character) image1_insensitive_filename (Character) image2_up_filename (Character) image2_down_filename (Character) image2_insensitive_filename (Character) item_select_type (Character) item_select_action (Character) item_link (Character) item_select_parameter (Character) item_menu_drop (Character) on_create_publish_event (Character) enable_rule (Character) disable_rule (Character) image_alternate_rule (Character) hide_rule (Character) item_control_type (Character) item_control_style (Character) substitute_text_property (Character) item_narration (Character) source_language_obj (Decimal)	instance_attribute_obj item_category_obj menu_item_obj product_module_obj

Table 7–2 gives details of the table’s indexes.

Table 7–2: gsm_menu_item index information

Index name	Elements	Type
XPKgsm_menu_item	menu_item_obj	Primary Unique
XAK1gsm_menu_item	menu_item_reference	Unique
XIE1gsm_menu_item	product_module_obj	Nonunique
XIE2gsm_menu_item	object_obj	Nonunique
XIE3gsm_menu_item	instance_attribute_obj	Nonunique
XIE4gsm_menu_item	item_category_obj	Nonunique
XIE5gsm_menu_item	menu_item_label	Nonunique
XIE6gsm_menu_item	menu_item_description	Nonunique
XIE7gsm_menu_item	security_token	Nonunique
XIE8gsm_menu_item	item_toolbar_label	Nonunique
XIE9gsm_menu_item	source_language_obj	Nonunique

7.3.2 gsc_instance_attribute table—gscia

This table contains instance attributes used in the application. Instance attributes change the behavior of generic objects. For example, an application has a generic object that behaves differently in creditor and debtor systems. When the application runs, an instance attribute of creditor or debtor is posted to determine its instance-specific functionality.

Table 7–3 lists the table’s FLA, fields, and foreign keys.

Table 7–3: gsc_instance_attribute table information

Table FLA	Fields (data type)	Foreign keys
gscia	instance_attribute_obj (Decimal) attribute_code (Character) attribute_description (Character) disabled (Logical) system_owned (Logical) attribute_type (Character)	instance_attribute_obj

Table 7–4 gives details of the table’s indexes.

Table 7–4: gsc_instance_attribute index information

Index name	Elements	Type
XPkgsc_instance_attribute	instance_attribute_obj	Primary Unique
XAK1gsc_instance_attribute	attribute_code	Unique
XAK2gsc_instance_attribute	attribute_type, attribute_code	Unique
XIE1gsc_instance_attribute	attribute_description	Nonunique

The instance attribute might be posted to the application by the menu option that launches the application or it might be hard-coded in the button that launches the application. For this reason, certain instance attributes are system-owned and cannot be maintained or deleted by users.

When security structures such as field security are set up, they might be defined globally, for a specific product, product module, or at an individual program level. The instance attribute is a level below the program level. It permits security settings for each instance of an application.

Instance attributes can be used for reporting to allow reports to be printed direct from menu options. The instance attribute code must map to the report_procedure_name in the report_definition table. Whenever a report definition is created, an instance attribute should automatically be created to facilitate this functionality.

7.3.3 gsc_item_category table—gscic

This table is used to categorize items into common groups. Typical groups might be ADM Navigation, ADM TableIO, or ADM Menu. Categories might also be used to group items into module-specific areas.

Table 7–5 lists the table’s FLA, fields, and foreign keys.

Table 7–5: gsc_item_category table information

Table FLA	Fields (data type)	Foreign keys
gscic	item_category_obj (Decimal) item_category_label (Character) item_category_description (Character) item_link (Character) system_owned (Logical) parent_item_category_obj (Decimal)	item_category_obj

Table 7–6 gives details of the table’s indexes.

Table 7–6: gsc_item_category index information

Index name	Elements	Type
XPKgsc_item_category	item_category_obj	Primary Unique
XAK1gsc_item_category	item_category_label	Unique
XAK2gsc_item_category	parent_item_category_obj item_category_label	Unique
XIE1gsc_item_category	item_category_description	Nonunique

7.3.4 gsm_menu_structure table—gsmms

This table defines the dynamic menu structures available. A menu structure must belong to a product, and if required, can be associated with a product module for sorting purposes.

The menu structure code is referenced in source code to build any dynamic menu items associated with the menu structure.

Table 7–7 lists the table’s FLA, fields, and foreign keys.

Table 7–7: `gsm_menu_structure` table information

Table FLA	Fields (data type)	Foreign keys
gsmms	menu_structure_obj (Decimal) product_module_obj (Decimal) product_obj (Decimal) menu_structure_code (Character) menu_structure_description (Character) disabled (Logical) system_owned (Logical) under_development (Logical) menu_item_obj (Decimal) menu_structure_type (Character) menu_structure_hidden (Logical) control_spacing (Integer) control_padding (Integer) menu_structure_narrative (Character)	menu_item_obj menu_structure_obj product_module_obj product_obj

Table 7–8 gives details of the table’s indexes.

Table 7–8: `gsm_menu_structure` index information

Index name	Elements	Type
XPKgsm_menu_structure	menu_structure_obj	Primary Unique
XAK1gsm_menu_structure	menu_structure_code	Unique
XAK2gsm_menu_structure	product_module_obj menu_structure_code	Unique
XIE2gsm_menu_structure	menu_structure_description	Nonunique
XIE3gsm_menu_structure	menu_structure_type	Nonunique
XIE4gsm_menu_structure	menu_item_obj	Nonunique

7.3.5 **gsm_menu_structure_item table—gsmmit**

This table associates menu items with menu structures. A menu structure can contain many menu items, and a menu item can be used by many menu structures.

This tables also defines the sequence that menu items appear within a menu structure.

Table 7–9 lists the table’s FLA, fields, and foreign keys.

Table 7–9: gsm_menu_structure_item table information

Table FLA	Fields (data type)	Foreign keys
gsmmit	menu_structure_obj (Decimal) menu_item_sequence (Integer) menu_item_obj (Decimal) child_menu_structure_obj (Decimal) menu_structure_item_obj (Decimal)	menu_item_obj menu_structure_obj

Table 7–10 gives details of the table’s indexes.

Table 7–10: gsm_menu_structure_item index information

Index name	Elements	Type
XPKgsm_menu_structure_item	menu_structure_obj menu_item_sequence	Primary Unique
XAK1gsm_menu_structure_item	menu_structure_item_obj	Unique
XIE1gsm_menu_structure_item	child_menu_structure_obj	Nonunique
XIE2gsm_menu_structure_item	menu_item_obj	Nonunique

7.3.6 **gsm_object_menu_structure table—gsmmom**

This table defines the dynamic menu structures, if any, used by the object. Only container objects can have dynamic menu structures.

If an instance attribute is specified, then the menu structure is dynamically built only if the instance attribute is passed in from the previous menu option. This aids pulling in different dynamic menu options for a specific object based on its use. For example, creditors and debtors might have different options.

Table 7–11 lists the table’s FLA, fields, and foreign keys.

Table 7–11: gsm_object_menu_structure table information

Table FLA	Fields (data type)	Foreign keys
gsmom	object_obj (Decimal) menu_structure_obj (Decimal) instance_attribute_obj (Decimal) object_menu_structure_obj (Decimal) menu_item_obj (Decimal) insert_submenu (Logical) menu_structure_sequence (Integer)	instance_attribute_obj menu_item_obj menu_structure_obj object_obj

Table 7–12 gives details of the table’s indexes.

Table 7–12: gsm_object_menu_structure index information

Index name	Elements	Type
XPKgsm_object_menu_structure	object_obj menu_structure_obj instance_attribute_obj	Primary Unique
XAK1gsm_object_menu_structure	menu_structure_obj object_obj instance_attribute_obj	Unique
XAK2gsm_object_menu_structure	object_menu_structure_obj	Unique
XIE1gsm_object_menu_structure	instance_attribute_obj	Nonunique
XIE2gsm_object_menu_structure	menu_item_obj object_obj instance_attribute_obj	Nonunique

7.3.7 **gsm_toolbar_menu_structure table—gsmtm**

This table is used to group bands or menu structures into a complete toolbar and menubar structure.

Table 7–13 lists the table’s FLA, fields, and foreign keys.

Table 7–13: gsm_toolbar_menu_structure table information

Table FLA	Fields (data type)	Foreign keys
gsmtm	object_obj (Decimal) menu_structure_sequence (Integer) menu_structure_obj (Decimal) toolbar_menu_structure_obj (Decimal) menu_structure_spacing (Integer) menu_structure_alignment (Character) menu_structure_row (Character) insert_rule (Logical)	menu_structure_obj object_obj

Table 7–14 gives details of the table’s indexes.

Table 7–14: gsm_toolbar_menu_structure index information

Index name	Elements	Type
XPKgsm_toolbar_menu_structure	object_obj menu_structure_sequence menu_structure_obj	Primary Unique
XAK1gsm_toolbar_menu_structur	toolbar_menu_structure_obj	Unique
XIE1gsm_toolbar_menu_structur	menu_structure_obj	Nonunique

7.3.8 **gsm_translated_menu_item table—gsmti**

This table stores menu item translations. Fields requiring translation are duplicated from the gsm_menu_item table and translations for them stored in this table by language.

This table is only required to be referenced if the user’s login language does not match the source language of the menu item, otherwise the standard fields on the gsm_menu_item can be used, that is, this table is only required if a translation is required.

Table 7–15 lists the table’s FLA, fields, and foreign keys.

Table 7–15: gsm_translated_menu_item table information

Table FLA	Fields (data type)	Foreign keys
gsmti	translated_menu_item_obj (Decimal) menu_item_obj (Decimal) source_language_obj (Decimal) language_obj (Decimal) menu_item_label (Character) tooltip_text (Character) alternate_shortcut_key (Character) item_toolbar_label (Character) image1_up_filename (Character) image1_down_filename (Character) image1_insensitive_filename (Character) image2_up_filename (Character) image2_down_filename (Character) image2_insensitive_filename (Character) item_narration (Character)	language_obj menu_item_obj

Table 7–16 gives details of the table’s indexes.

Table 7–16: gsm_translated_menu_item index information

Index name	Elements	Type
XPKgsm_translated_menu_item	translated_menu_item_obj	Primary Unique
XAK1gsm_translated_menu_item	menu_item_obj language_obj	Unique
XIE1gsm_translated_menu_item	source_language_obj	Nonunique
XIE2gsm_translated_menu_item	language_obj	Nonunique

Module Group

This chapter covers the group of tables in the Progress Dynamics Repository that store information about product modules. This chapter covers the following topics:

- [Module group structure](#)
- [Table descriptions](#)

8.1 Module group structure

The Module group stores information about installed products and product modules. [Figure 8–1](#) shows the structure and relationships of the tables in this group.

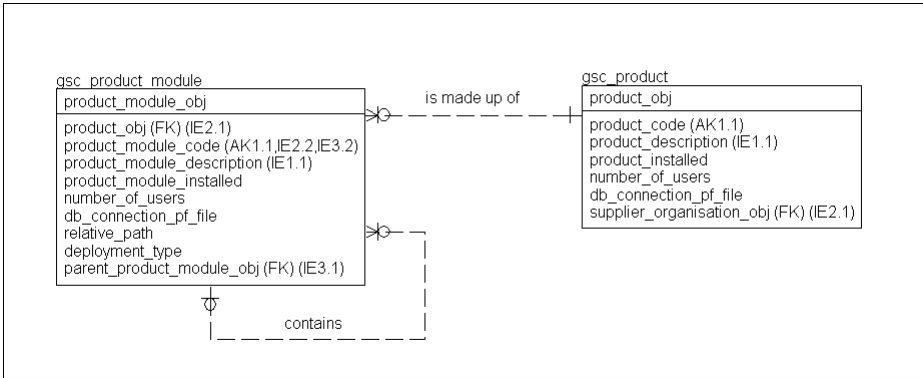


Figure 8–1: Module group

8.2 Table descriptions

The following sections provide detailed information about each table in the Module group. The Module group contains the following tables:

- [gsc_product table—gscpr](#)
- [gsc_product_module table—gscpm](#)

8.2.1 gsc_product table—gscpr

This table contains information about the installed products with appropriate license details.

[Table 8–1](#) lists the table’s FLA, fields, and foreign keys.

Table 8–1: gsc_product table information

Table FLA	Fields (data type)	Foreign keys
gscpr	product_obj (Decimal) product_code (Character) product_description (Character) product_installed (Logical) number_of_users (Integer) db_connection_pf_file (Character) supplier_organisation_obj (Decimal)	product_code product_obj

[Table 8–2](#) gives details of the table’s indexes.

Table 8–2: gsc_product index information

Index name	Elements	Type
XPkgsc_product	product_obj	Primary Unique
XAK1gsc_product	product_code	Unique
XIE1gsc_product	product_description	Nonunique
XIE2gsc_product	supplier_organisation_obj	Nonunique

8.2.2 gsc_product_module table—gscpm

This table contains information about the installed product modules with appropriate license details. The table includes a recursive join to support product module hierarchies.

Table 8–3 lists the table’s FLA, fields, and foreign keys.

Table 8–3: gsc_product_module table information

Table FLA	Fields (data type)	Foreign keys
gscpm	product_module_obj (Decimal) product_obj (Decimal) product_module_code (Character) product_module_description (Character) product_module_installed (Logical) number_of_users (Integer) db_connection_pf_file (Character) relative_path (Character) deployment_type (Character) parent_product_module_obj (Decimal)	product_module_code product_module_obj product_obj

Table 8–4 gives details of the table’s indexes.

Table 8–4: gsc_product_module index information

Index name	Elements	Type
XPKgsc_product_module	product_module_obj	Primary Unique
XAK1gsc_product_module	product_module_code	Unique
XIE1gsc_product_module	product_module_description	Nonunique
XIE2gsc_product_module	product_obj product_module_code	Nonunique
XIE3gsc_product_module	parent_product_module_obj product_module_code	Nonunique

Multi-media and Comment Group

This chapter covers the group of tables in the Progress Dynamics Repository that store information about multi-media and comments. This chapter covers the following topics:

- [Multi-media and Comment group structure](#)
- [Table descriptions](#)

9.1 Multi-media and Comment group structure

The Multi-media and Comment group stores information about multi-media and generic comment files that can be linked to some entity in your application. [Figure 9–1](#) shows the structure and relationships of the tables in this group.

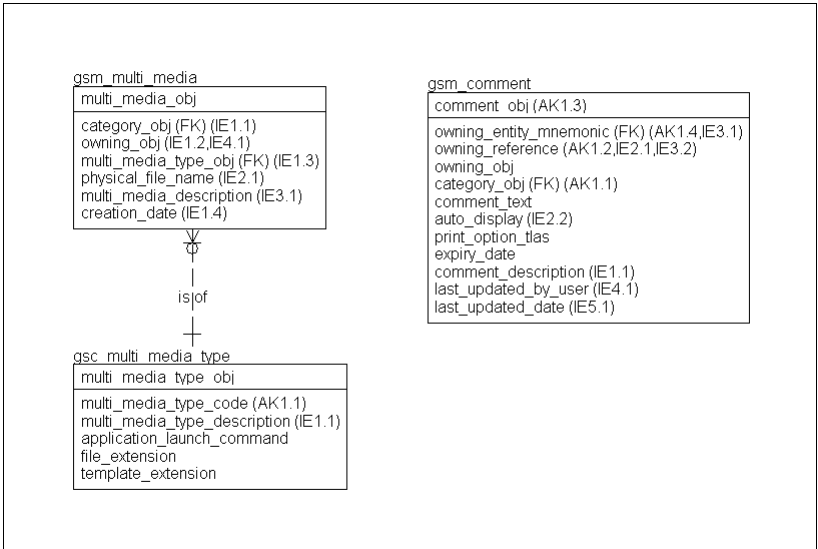


Figure 9–1: Multi-media and Comment group

9.2 Table descriptions

The following sections provide detailed information about each table in the Multi-media and Comment group. The Multi-media and Comment group contains the following tables:

- [gsc_multi_media_type table—gscmm](#)
- [gsm_comment table—gsmcm](#)
- [gsm_multi_media table—gsmmm](#)

9.2.1 gsc_multi_media_type table—gscmm

This table contains information related to different types of multi-media files.

[Table 9–1](#) lists the table’s FLA, fields, and foreign keys.

Table 9–1: gsc_multi_media_type table information

Table FLA	Fields (data type)	Foreign keys
gscmm	multi_media_type_obj (Decimal) multi_media_type_code (Character) multi_media_type_description (Character) application_launch_command (Character) file_extension (Character) template_extension (Character)	multi_media_type_obj

[Table 9–2](#) gives details of the table’s indexes.

Table 9–2: gsc_multi_media_type index information

Index name	Elements	Type
XPKgsc_multi_media_type	multi_media_type_obj	Primary Unique
XAK1gsc_multi_media_type	multi_media_type_code	Unique
XIE1gsc_multi_media_type	multi_media_type_description	Nonunique

9.2.2 gsm_comment table—gsmcm

This table lists generic comments that can be linked to any entity.

[Table 9–3](#) lists the table’s FLA, fields, and foreign keys.

Table 9–3: gsm_comment table information

Table FLA	Fields (data type)	Foreign keys
gsmcm	comment_obj (Decimal) owning_entity_mnemonic (Character) owning_reference (Character) owning_obj (Decimal) category_obj (Decimal) comment_text (Character) auto_display (Logical) print_option_tlas (Character) expiry_date (Date) comment_description (Character) last_updated_by_user (Character) last_updated_date (Date)	category_obj owning_obj

Table 9–4 gives details of the table’s indexes.

Table 9–4: gsm_comment index information

Index name	Elements	Type
XPkgsm_comment	comment_obj	Primary Unique
XAK1gsm_comment	category_obj owning_reference comment_obj owning_entity_mnemonic	Unique
XIE1gsm_comment	comment_description	Nonunique
XIE2gsm_comment	owning_reference auto_display	Nonunique
XIE3gsm_comment	owning_entity_mnemonic owning_reference	Nonunique
XIE4gsm_comment	last_updated_by_user	Nonunique
XIE5gsm_comment	last_updated_date	Nonunique

9.2.3 gsm_multi_media table—gsmmm

This table stores information on generic multi-media files that can be linked to any entity.

Table 9–5 lists the table’s FLA, fields, and foreign keys.

Table 9–5: gsm_multi_media table information

Table FLA	Fields (data type)	Foreign keys
gsmmm	multi_media_obj (Decimal) category_obj (Decimal) owning_obj (Decimal) multi_media_type_obj (Decimal) physical_file_name (Character) multi_media_description (Character) creation_date (Date)	category_obj multi_media_type_obj owning_obj

Table 9–6 gives details of the table’s indexes.

Table 9–6: gsm_multi_media index information

Index name	Elements	Type
XPKgsm_multi_media	multi_media_obj	Primary Unique
XIE1gsm_multi_media	category_obj owning_obj multi_media_type_obj creation_date	Nonunique
XIE2gsm_multi_media	physical_file_name	Nonunique
XIE3gsm_multi_media	multi_media_description	Nonunique
XIE4gsm_multi_media	owning_obj	Nonunique

Object Group

This chapter covers the group of tables in the Progress Dynamics Repository that store information about the objects in your application. This chapter covers the following topics:

- [Object group structure](#)
- [Managers](#)
- [Table descriptions](#)

10.1 Object group structure

The Object group is the most extensive set of tables in the Repository. The main table in the Object group is ryc_smartobject. Because various objects relate to each other, records on ryc_smartobject might reference other records on the table. [Figure 10–1](#) shows these possible relationships.

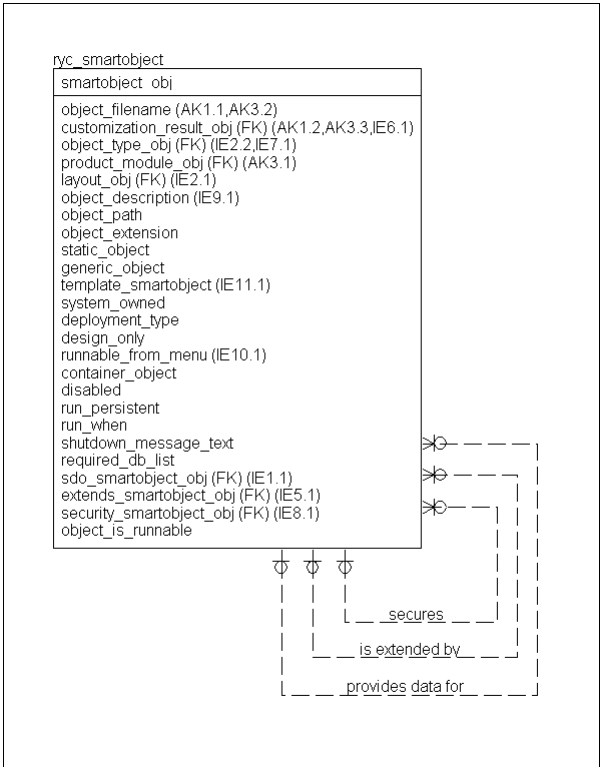


Figure 10–1: ryc_smartobject table

10.1.1 Main tables in Object group

The main subgroup of tables is built around the `ryc_smartobject` table. This main subgroup is supplemented by supporting tables that contain static definitions.

Figure 10–2 shows structure and relationships of the tables in the main subgroup.

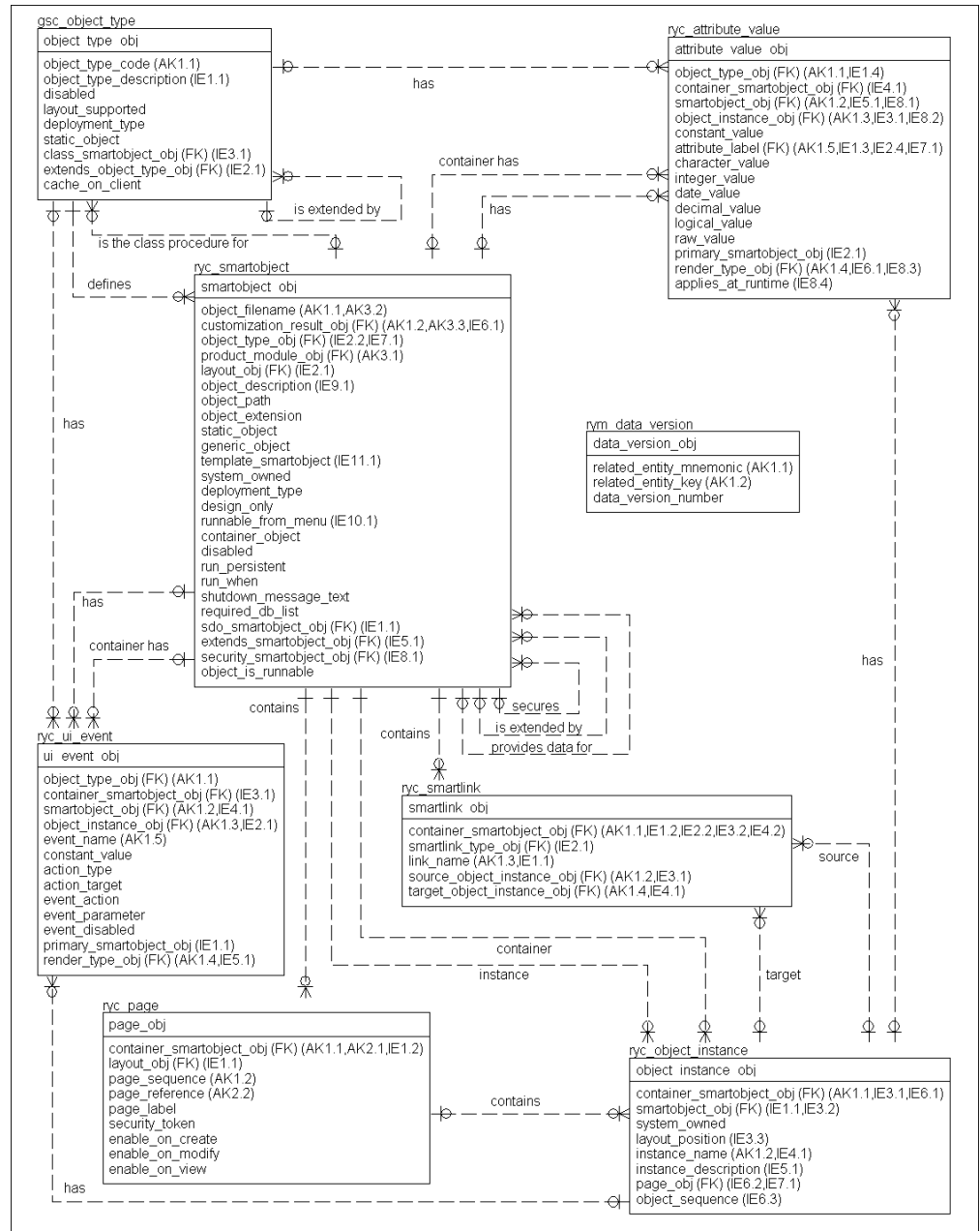


Figure 10–2: Main tables in Object group

10.1.2 Static definition tables

The Object group includes several tables that supply static definitions to the other tables in the group. These tables supply definitions to support object attributes, Progress SmartLinks™, and page layouts.

As shown in [Figure 10–3](#), the ryc_attribute_value table is supported by attribute definitions from two supporting tables.

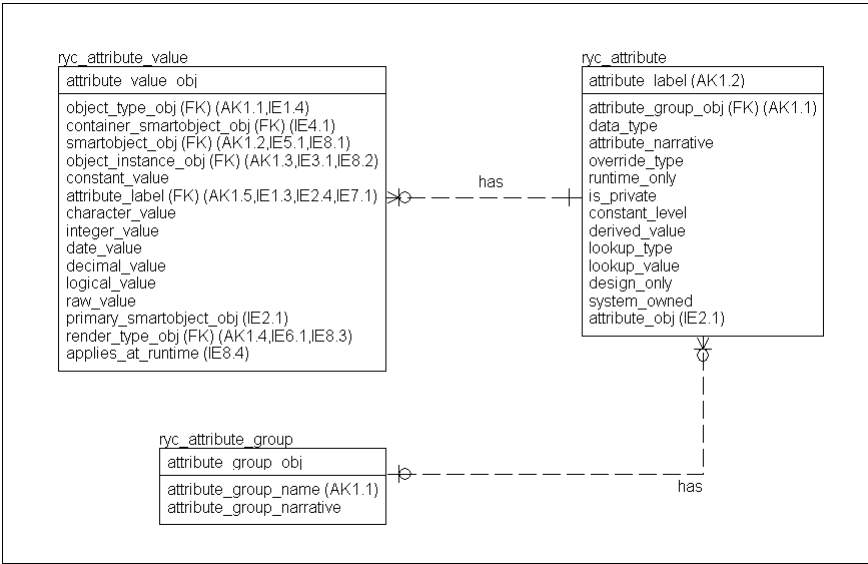


Figure 10–3: Attribute static definition tables

As shown in [Figure 10–4](#), the gsc_object_type table and the ryc_smartlink table are also supported by SmartLink definitions taken from two tables.

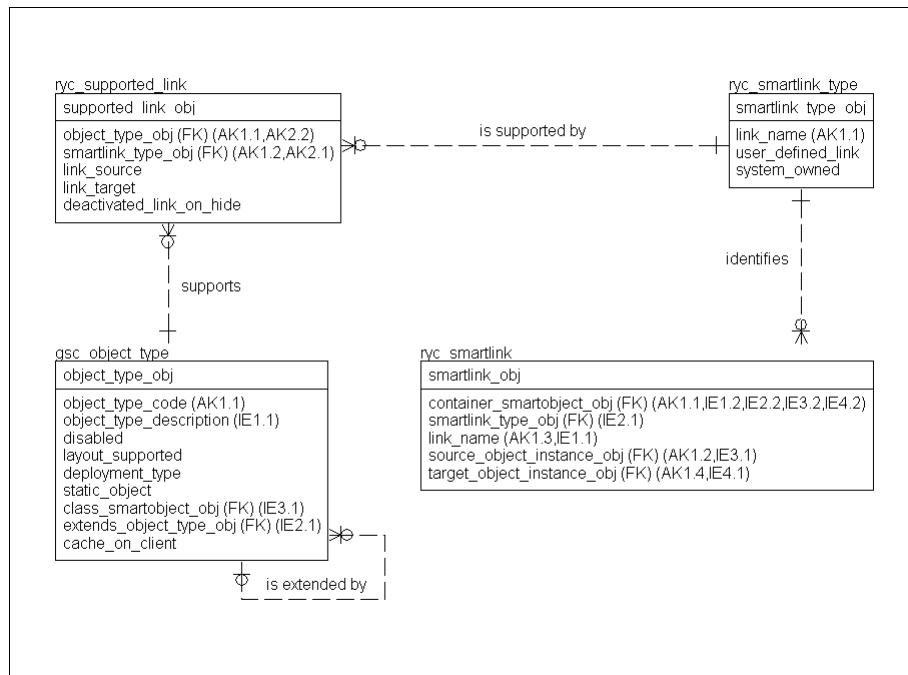


Figure 10–4: SmartLink static definition tables

Finally, as shown in [Figure 10–5](#), the ryc_smartobject table and the ryc_page table are supported by page layout definitions from the ryc_layout table.

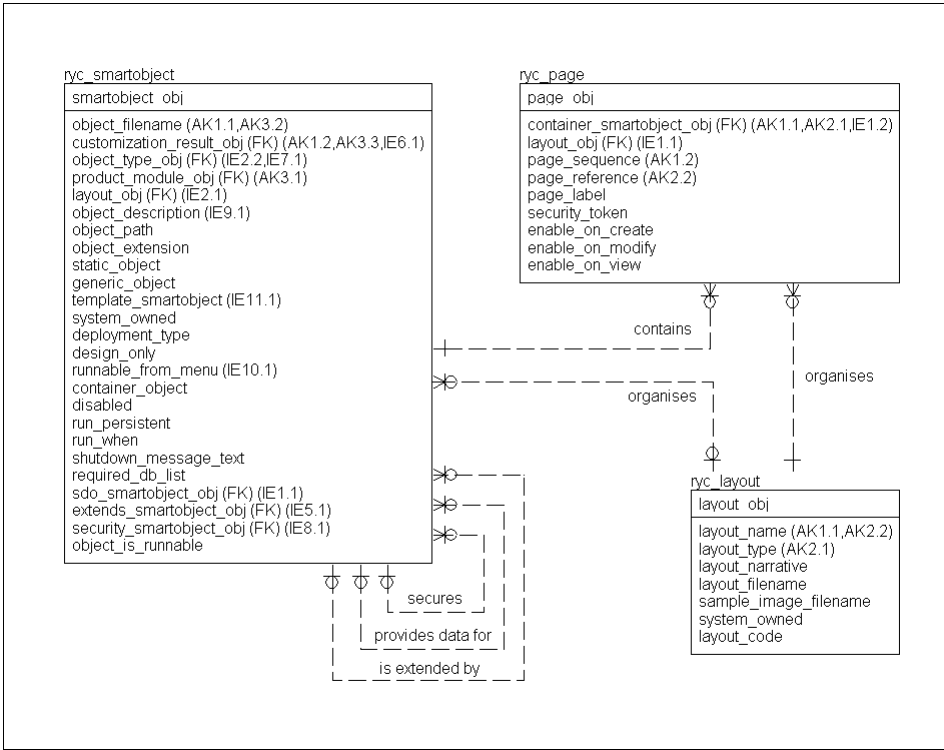


Figure 10–5: Page layout static definition tables

Figure 10–7 shows the relationships between the tables for defining object attribute values.

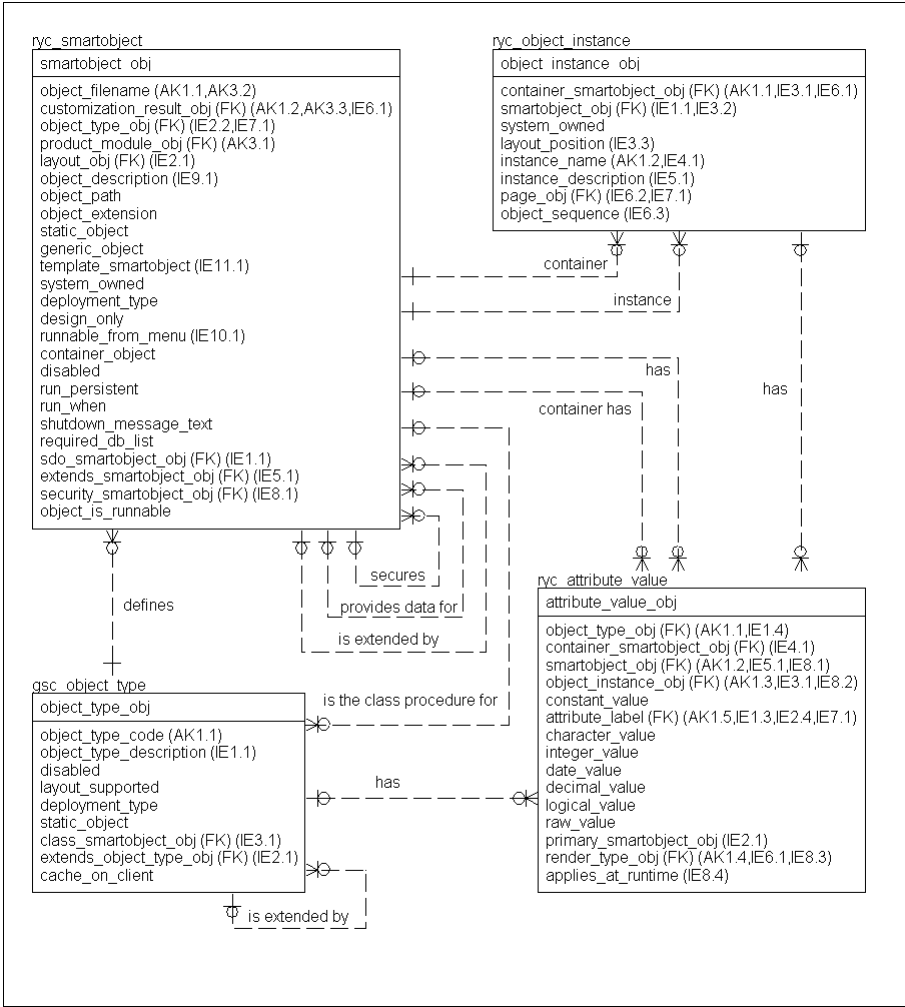


Figure 10–7: Table relationships for object attribute values

10.2 Managers

The Repository Managers use these tables to provide the Progress Dynamics framework with information to build application user interfaces. For more detail about the temp-tables and APIs the Repository Managers use to manipulate data from the Menu and Toolbar group tables, see the chapter on the Repository Managers in the *Progress Dynamics Managers API Reference*.

10.3 Table descriptions

The following sections provide detailed information about each table in the Object group. The Object group contains the following tables:

- [ryc_smartobject table—rycso](#)
- [gsc_object_type table—gscot](#)
- [ryc_attribute table—rycat](#)
- [ryc_attribute_group table—rycap](#)
- [ryc_attribute_value table—rycav](#)
- [ryc_layout table—rycla](#)
- [ryc_object_instance table—rycoi](#)
- [ryc_page table—rycpa](#)
- [ryc_smartlink table—rycsm](#)
- [ryc_smartlink_type table—rycst](#)
- [ryc_supported_link table—rycsl](#)
- [ryc_ui_event table—rycue](#)
- [rym_data_version table—rycdv](#)

10.3.1 ryc_smartobject table—rycso

The ryc_smartobject table is the central table of the Object group. This table lists every object known to the Repository, whether static or dynamic. You should register in the Repository every object that can be run in your application, including visual objects and business logic procedures. You should also register items like the images used on buttons. The only files not registered in the Repository are include files.

If a file is registered, it can be added as an instance to containers, added to menus, have security applied to it, be run as part of flows and events, and be set up for automated deployment.

Table 10–1 lists the table’s FLA, fields, and foreign keys.

Table 10–1: ryc_smartobject table information

Table FLA	Fields (data type)	Foreign keys
rycso	smartobject_obj (Decimal) object_filename (Character) customization_result_obj (Decimal) object_type_obj (Decimal) product_module_obj (Decimal) layout_obj (Decimal) object_description (Character) object_path (Character) object_extension (Character) static_object (Logical) generic_object (Logical) template_smartobject (Logical) system_owned (Logical) deployment_type (Character) design_only (Logical) runnable_from_menu (Logical) container_object (Logical) disabled (Logical) run_persistent (Logical) run_when (Character) shutdown_message_text (Character) required_db_list (Character) sdo_smartobject_obj (Decimal) extends_smartobject_obj (Decimal) security_smartobject_obj (Decimal) object_is_runnable (Logical)	customization_result_obj layout_obj object_type_obj product_module_obj smartobject_obj

Table 10–2 gives details of the table’s indexes.

Table 10–2: ryc_smartobject index information

Index name	Elements	Type
XPKryc_smartobject	smartobject_obj	Primary Unique
XAK1ryc_smartobject	object_filename customization_result_obj	Unique
XAK3ryc_smartobject	product_module_obj object_filename customization_result_obj	Unique
XIE1ryc_smartobject	sdo_smartobject_obj	Nonunique
XIE2ryc_smartobject	layout_obj object_type_obj	Nonunique
XIE5ryc_smartobject	extends_smartobject_obj	Nonunique
XIE6ryc_smartobject	customization_result_obj	Nonunique
XIE7ryc_smartobject	object_type_obj	Nonunique
XIE8ryc_smartobject	security_smartobject_obj	Nonunique
XIE9ryc_smartobject	object_description	Nonunique
XIE10ryc_smartobject	runnable_from_menu	Nonunique
XIE11ryc_smartobject	template_smartobject	Nonunique

Objects are further defined in the many child tables in the Object group. The child tables might or might not apply to a certain objects based on the type of object. For example, information regarding links, pages, and instances only pertain to objects that are containers. Even in a certain type of object, every object might not use the same child tables. For example, viewers, browsers, and SDOs are containers for datafields. But they do not use pages. Only window containers support pages.

In Version 1.1 Repository, object properties were split between this table and another one, `gsc_object`. To improve performance, the Version 2 Repository stores all those properties on this table and drops the `gsc_object` table. Because of this, not every property on this table applies to every object, and there might be some redundant properties.

NOTES:

- For data versioning, the delete trigger's referential integrity has been turned off because it would prevent an object being deleted if it is used on some container. This allows imports and object assignments to work. Therefore, the delete trigger referential integrity must be manually coded where it is required.
- The automatic delete cascade of SmartObject attributes has been turned off. It might delete attributes for instances of a SmartObject when it should not delete them. A specific delete trigger customization was added that joins on the `primary_smartobject_obj` in the `ryc_attribute_value` table when deleting attributes. This ensures that only attributes for the SmartObject are deleted, and not attributes for instances of the SmartObject.
- This table can optionally support customization using custom result codes. The unique key to this table is made up of the object filename and a result code. This allows the same object name to have multiple result codes attached to it. Only the custom objects created in this manner contain the newly added or overridden behavior. The master default object has a result code of 0 and must exist for every object.

This allows you to customize the information on this table and any of its child tables for objects that have custom result codes assigned. When you read a SmartObject based on the filename, the result code must be specified, using 0 if you are looking for the master.

- The recursive join with a rolename of `extends_smartobject_obj` supports development of new functionality. It is not yet used in the framework. to support inheritance.

10.3.2 gsc_object_type table—gscot

This table defines the types of programs supported. A record must exist for the various support templates, such as Object Controller, Menu Controller, Progress SmartFolder™, Progress SmartBrowser™, Progress SmartViewer™, and Progress SmartDataObjects™.

When objects are created, they must be assigned an object type.

Table 10–3 lists the table’s FLA, fields, and foreign keys.

Table 10–3: gsc_object_type table information

Table FLA	Fields (data type)	Foreign keys
gscot	object_type_obj (Decimal) object_type_code (Character) object_type_description (Character) disabled (Logical) layout_supported (Logical) deployment_type (Character) static_object (Logical) class_smartobject_obj (Decimal) extends_object_type_obj (Decimal) cache_on_client (Logical) custom_object_type_obj (Decimal)	object_type_obj

Table 10–4 gives details of the table’s indexes.

Table 10–4: gsc_object_type index information

Index name	Elements	Type
XPKgsc_object_type	object_type_obj	Primary Unique
XAK1gsc_object_type	object_type_code	Unique
XIE1gsc_object_type	object_type_description	Nonunique
XIE2gsc_object_type	extends_object_type_obj	Nonunique
XIE3gsc_object_type	class_smartobject_obj	Nonunique
XIE4gsc_object_type	custom_object_type_obj	Nonunique

The object type is used as a grouping mechanism for security. This enables restrictions to be created for certain types of objects, rather than having to set up security for every object.

A recursive join exists for the object type to facilitate definition of object type hierarchies (class hierarchies). This is useful for attribute inheritance at multiple levels of object type. For example, an object type could be defined for a fill-in, then a child of this might be an integer fill-in, and a child of that might be an object id.

To support customizations for object types (classes), you can specify a `custom_object_type_obj` to identify an additional class structure to include as part of another class, such as to add additional custom attributes and class super procedures. This makes it easier to add customizations to a class in the middle of the hierarchy without having to modify the hierarchy itself. You would not have to change the `extends_object_type_obj` of the subclasses to point at the new custom class.

A specified custom class is added below the customized class. If you customize the data class with a custom class called “datacustom,” then at run time datacustom is added as a subclass of data. It overrides and extends the behavior of all objects that further extend the data class. This also means that all classes below the data class would also automatically extend the custom class datacustom rather than data. So datacustom ends up in the middle of the class hierarchy.

The custom class itself can include a structure below it to provide multiple levels of customization. All of the customizations are inserted below the class, and classes that previously extended the class will instead extend the class at the bottom of the customized classes. For example, if the custom class datacustom included an additional subclass, datacustom2 that extends datacustom, then the subclasses of the original class, data, point at datacustom2, which points at datacustom, which then points back at the original class, data.

The field `custom_object_type_obj` is specified in the deployment datasets exclude field list so that it is never overwritten as part of deploying changes to the class structure. This avoids the loss of the customizations with future upgrades.

10.3.3 ryc_attribute table—rycat

This table defines the attributes that can be allocated to objects, such as size, position, window title, query, and the WHERE clause. The attributes are used to define the properties of dynamic objects, and to dynamically alter the behavior of static objects.

[Table 10–5](#) lists the table’s FLA, fields, and foreign keys.

Table 10–5: ryc_attribute table information

Table FLA	Fields (data type)	Foreign keys
rycat	attribute_label (Character) attribute_group_obj (Decimal) data_type (Integer) attribute_narrative (Character) override_type (Character) runtime_only (Logical) is_private (Logical) constant_level (Character) derived_value (Logical) lookup_type (Character) lookup_value (Character) design_only (Logical) system_owned (Logical) attribute_obj (Decimal)	attribute_group_obj attribute_label

[Table 10–6](#) gives details of the table’s indexes.

Table 10–6: ryc_attribute index information

Index name	Elements	Type
XPKryc_attribute	attribute_label	Primary Unique
XAK1ryc_attribute	attribute_group_ob attribute_label	Unique
XIE2ryc_attribute	attribute_obj	Nonunique

Certain attributes are required for the framework to function correctly. These are set to system-owned to prevent accidental deletion. Only users classified as able to maintain system-owned information can manipulate this data. In most cases, the actual attribute label needs to match to a valid Progress-supported attribute.

Because of the usefulness of allowing attributes to be defined at various levels, most dynamic data for SmartObjects uses these attributes.

Browser query, sort order, and WHERE clauses use these attributes. Other functions that use attributes include creating container window titles, deciding which window to run based on various button actions in a browser, determining status bar configurations, page enabling and disabling, field enabling and disabling by object instance, and deciding which toolbar items are included in the menu.

10.3.4 ryc_attribute_group table—rycap

This table aids the logical grouping of attributes to simplify their use. Sample groupings might be geometry and status bar. The primary use of this table is to present the attributes to the user in a more effective and usable manner. Attribute groups support Progress Dynamics tools used only at design time. They serve no purpose at run time.

Table 10–7 lists the table’s FLA, fields, and foreign keys.

Table 10–7: ryc_attribute_group table information

Table FLA	Fields (data type)	Foreign keys
rycap	attribute_group_obj (Decimal) attribute_group_name (Character) attribute_group_narrative (Character)	attribute_group_obj

Table 10–8 gives details of the table’s indexes.

Table 10–8: ryc_attribute_group index information

Index name	Elements	Type
XPKryc_attribute_group	attribute_group_obj	Primary Unique
XAKIryc_attribute_group	attribute_group_name	Unique

10.3.5 **ryc_attribute_value table—rycav**

This table associates attributes with object types, SmartObjects, and SmartObject instances. It also specifies the value of the attribute in the appropriate native data type. If a native data type is not available, the CHARACTER data type is used.

The list of attribute values defined for the object type (class) must be complete. Unless an attribute is defined at the class level, it cannot be set within a subclass. Attribute values are not cascaded down to subclasses, and only override entries exist at subclass levels. To retrieve all the attribute values for an object, you must also read the attributes for all parent classes. Records only exist for subclasses if the value has been specifically overridden for that subclass.

When creating entries in this table for attributes associated with an object type, the object numbers for the SmartObject and instance are 0.

When creating entries in the table for a SmartObject, the object type field is populated to avoid having 0 in the key. When creating attributes for an object instance, both the object type and the SmartObject are populated. This ensures effective use of the alternate keys.

NOTE: When you look for the attributes for an object type, check for the specific object type and 0 values for the SmartObject and instance fields.

When multiple rendering engines are supported and used, the `render_type_object` adds another dimension to the possible attribute values. You can specify a rendering engine type for attributes at the class, master, and instance levels. If an attribute is specified at the class level for a specific rendering engine type, and not for a 0 rendering engine type, then that attribute is only used for the specific rendering engine type and is not applied across all rendering engines.

Table 10–9 lists the table’s FLA, fields, and foreign keys.

Table 10–9: ryc_attribute_value table information

Table FLA	Fields (data type)	Foreign keys
rycav	attribute_value_obj (Decimal) object_type_obj (Decimal) container_smartobject_obj (Decimal) smartobject_obj (Decimal) object_instance_obj (Decimal) constant_value (Logical) attribute_label (Character) character_value (Character) integer_value (Integer) date_value (Date) decimal_value (Decimal) logical_value (Logical) raw_value (Raw) primary_smartobject_obj (Decimal) render_type_obj (Decimal) applies_at_runtime (Logical)	attribute_label object_instance_obj object_type_obj render_type_obj smartobject_obj

Table 10–10 gives details of the table’s indexes.

Table 10–10: ryc_attribute_value index information

Index name	Elements	Type
XPKryc_attribute_value	attribute_value_obj	Primary Unique
XAK1ryc_attribute_value	object_type_obj smartobject_obj object_instance_obj render_type_obj attribute_label	Unique
XIE1ryc_attribute_value	attribute_label object_type_obj	Nonunique
XIE2ryc_attribute_value	primary_smartobject_obj attribute_label	Nonunique
XIE3ryc_attribute_value	object_instance_obj	Nonunique
XIE4ryc_attribute_value	container_smartobject_obj	Nonunique
XIE5ryc_attribute_value	smartobject_obj	Nonunique
XIE6ryc_attribute_value	render_type_obj	Nonunique
XIE7ryc_attribute_value	attribute_label	Nonunique
XIE8ryc_attribute_value	smartobject_obj object_instance_obj render_type_obj applies_at_runtime	Nonunique

10.3.6 ryc_layout table—rycla

This table defines the available page layouts for pages on SmartFolder windows, such as one browser with a toolbar underneath, a set number of viewers above each other, two side-by-side viewers, or two side-by-side browsers. It also defines the available frame layouts for objects on a frame, such as how many columns exist.

This table specifies the program that is responsible for the layout when the window or frame is constructed or resized.

As of Progress Dynamics Version 2, only relative layouts are fully supported. Relative layouts have a layout code of 6.

[Table 10–11](#) lists the table’s FLA, fields, and foreign keys.

Table 10–11: ryc_layout table information

Table FLA	Fields (data type)	Foreign keys
rycla	layout_obj (Decimal) layout_name (Character) layout_type (Character) layout_narrative (Character) layout_filename (Character) sample_image_filename (Character) system_owned (Logical) layout_code (Character)	layout_obj

[Table 10–12](#) gives details of the table’s indexes.

Table 10–12: ryc_layout index information

Index name	Elements	Type
XPKrxc_layout	layout_obj	Primary Unique
XAK1rxc_layout	layout_name	Unique
XAK2rxc_layout	layout_type layout_name	Unique

10.3.7 ryc_object_instance table—rycoi

This table lists the running instances of an object on a container. This aids in the allocation of specific attributes, links, and page numbers for the specific instance of an object.

The instance name must be unique within a container. It is used to manage and locate instances of objects on a container and apply customizations to the same instance.

This table also defines the container page on which the instance appears and, where applicable, the order of the objects within a page.

Table 10–13 lists the table’s FLA, fields, and foreign keys.

Table 10–13: ryc_object_instance table information

Table FLA	Fields (data type)	Foreign keys
rycoi	object_instance_obj (Decimal) container_smartobject_obj (Decimal) smartobject_obj (Decimal) system_owned (Logical) layout_position (Character) instance_name (Character) instance_description (Character) page_obj (Decimal) object_sequence (Integer)	object_instance_obj page_obj smartobject_obj

Table 10–14 gives details of the table’s indexes.

Table 10–14: ryc_object_instance index information

Index name	Elements	Type
XPKryc_object_instance	object_instance_obj	Primary Unique
XAK1ryc_object_instance	container_smartobject_obj instance_name	Unique
XIE1ryc_object_instance	smartobject_obj	Nonunique
XIE3ryc_object_instance	container_smartobject_obj smartobject_obj layout_position	Nonunique
XIE4ryc_object_instance	instance_name	Nonunique
XIE5ryc_object_instance	instance_description	Nonunique
XIE6ryc_object_instance	container_smartobject_obj page_obj object_sequence	Nonunique
XIE7ryc_object_instance	page_obj	Nonunique

10.3.8 ryc_page table—rycpa

This table defines the actual pages in a container. All containers have at least one page, page 0. Page 0 and all objects on it are always displayed. If there are no other pages, then a tab folder is not visualized.

Table 10–15 lists the table’s FLA, fields, and foreign keys.

Table 10–15: ryc_page table information

Table FLA	Fields (data type)	Foreign keys
rycpa	page_obj (Decimal) container_smartobject_obj (Decimal) layout_obj (Decimal) page_sequence (Integer) page_reference (Character) page_label (Character) security_token (Character) enable_on_create (Logical) enable_on_modify (Logical) enable_on_view (Logical)	layout_obj page_obj

Table 10–16 gives details of the table’s indexes.

Table 10–16: ryc_page index information

Index name	Elements	Type
XPKryc_page	page_obj	Primary Unique
XAK1ryc_page	container_smartobject_obj page_sequence	Unique
XAK2ryc_page	container_smartobject_obj page_reference	Unique
XIE1ryc_page	layout_obj container_smartobject_obj	Nonunique

10.3.9 ryc_smartlink table—rycsm

This table defines the actual SmartLinks between objects on a container to aid in object communication. The link name can be user-defined or automatically copied from the SmartLink type for system-supported links.

If the source object instance is not specified, then the source defaults to the container. If the target object instance is not specified, then the target defaults to the container.

[Table 10–17](#) lists the table's FLA, fields, and foreign keys.

Table 10–17: ryc_smartlink table information

Table FLA	Fields (data type)	Foreign keys
rycsm	smartlink_obj (Decimal) container_smartobject_obj (Decimal) smartlink_type_obj (Decimal) link_name (Character) source_object_instance_obj (Decimal) target_object_instance_obj (Decimal)	None

Table 10–18 gives details of the table’s indexes.

Table 10–18: ryc_smartlink index information

Index name	Elements	Type
XPKryc_smartlink	smartlink_obj	Primary Unique
XAK1ryc_smartlink	container_smartobject_obj source_object_instance_obj link_name target_object_instance_obj	Unique
XIE1ryc_smartlink	link_name container_smartobject_obj	Nonunique
XIE2ryc_smartlink	smartlink_type_obj container_smartobject_obj	Nonunique
XIE3ryc_smartlink	source_object_instance_obj container_smartobject_obj	Nonunique
XIE4ryc_smartlink	target_object_instance_obj container_smartobject_obj	Nonunique

10.3.10 ryc_smartlink_type table—rycst

This table defines the supported SmartLinks available for linking objects on containers for object communication purposes.

Table 10–19 lists the table’s FLA, fields, and foreign keys.

Table 10–19: ryc_smartlink_type table information

Table FLA	Fields (data type)	Foreign keys
rycst	smartlink_type_obj (Decimal) link_name (Character) user_defined_link (Logical) system_owned (Logical)	smartlink_type_obj

Table 10–20 gives details of the table’s indexes.

Table 10–20: ryc_smartlink_type index information

Index name	Elements	Type
XPKryc_smartlink_type	smartlink_type_obj	Primary Unique
XAKlryc_smartlink_type	link_name	Unique

This table provides a valid list of SmartLinks to choose from when building generic containers. Additional SmartLinks can be implemented by creating a user-defined link.

When the SmartLink is not a user-defined link, the actual link name is cascaded down onto the ryc_smartlink table.

The ryc_supported_link table is used to highlight the types of links that are expected between two SmartObjects.

10.3.11 ryc_supported_link table—rycsl

This table defines the supported SmartLinks for the various types of SmartObjects. It lists whether the link can be a source, a target, or both. If a SmartObject does not support links, then there are no entries in this table for that SmartObject.

Table 10–21 lists the table’s FLA, fields, and foreign keys.

Table 10–21: ryc_supported_link table information

Table FLA	Fields (data type)	Foreign keys
rycsl	supported_link_obj (Decimal) object_type_obj (Decimal) smartlink_type_obj (Decimal) link_source (Logical) link_target (Logical) deactivated_link_on_hide (Logical)	object_type_obj smartlink_type_obj

Table 10–22 gives details of the table’s indexes.

Table 10–22: ryc_supported_link index information

Index name	Elements	Type
XPKryc_supported_link	supported_link_obj	Primary Unique
XAK1ryc_supported_link	object_type_obj smartlink_type_obj	Unique
XAK2ryc_supported_link	smartlink_type_obj object_type_obj	Unique

User-defined links should not be set up in this table. This table ensures that when objects on containers are linked, only valid system and user-defined links are used. This table is a developer’s aid.

10.3.12 ryc_ui_event table—rycue

This table stores information about user interface (UI) events for a SmartObject. It works like the ryc_attribute_value table. UI events can be associated with object types, SmartObjects, and SmartObject instances. This allows the attachment of UI events to dynamic objects.

Table 10–23 lists the table’s FLA, fields, and foreign keys.

Table 10–23: ryc_ui_event table information

Table FLA	Fields (data type)	Foreign keys
rycue	ui_event_obj (Decimal) object_type_obj (Decimal) container_smartobject_obj (Decimal) smartobject_obj (Decimal) object_instance_obj (Decimal) event_name (Character) constant_value (Logical) action_type (Character) action_target (Character) event_action (Character) event_parameter (Character) event_disabled (Logical) primary_smartobject_obj (Decimal) render_type_obj (Decimal)	object_instance_obj object_type_obj render_type_obj smartobject_obj

Table 10–24 gives details of the table’s indexes.

Table 10–24: ryc_ui_event index information

Index name	Elements	Type
XPKryc_ui_event	ui_event_obj	Primary Unique
XAK1ryc_ui_event	object_type_obj smartobject_obj object_instance_obj render_type_obj event_name	Unique
XIE1ryc_ui_event	primary_smartobject_obj	Nonunique
XIE2ryc_ui_event	object_instance_obj	Nonunique
XIE3ryc_ui_event	container_smartobject_obj	Nonunique
XIE4ryc_ui_event	smartobject_obj	Nonunique
XIE5ryc_ui_event	render_type_obj	Nonunique

When creating entries in this table for events associated with an object type, the object numbers for the SmartObject and instance are 0.

When creating entries in the table for a SmartObject, the object type field is populated to avoid having 0 in the key. When creating events for an object instance, both the object type and the SmartObject are populated. This ensures effective use of the alternate keys.

NOTE: When you look for the attributes for an object type, check for the specific object type and 0 values for the SmartObject and instance fields.

Events are not cascaded down to subclasses. Entries only exist at subclass levels for overrides. To retrieve all the events for an object, you must also read the events for all the object’s parent classes.

When multiple rendering engines are supported and used, the render_type_object adds another dimension to the possible events. You can specify a rendering engine type for events at the class, master, and instance levels. If an event is specified at the class level for a specific rendering engine type, and not for a 0 rendering engine type, then that event is only used for the specific rendering engine type and is not applied across all rendering engines.

10.3.13 rym_data_version table—rycdv

This table aids in the generic storage of data version numbers without having to add a specific version number field to tables that require version control. It is used in the context of versioning SmartObjects, but might also be used to record a version number for any data, such as menu items or help.

Table 10–25 lists the table’s FLA, fields, and foreign keys.

Table 10–25: rym_data_version table information

Table FLA	Fields (data type)	Foreign keys
rymdv	data_version_obj (Decimal) related_entity_mnemonic (Character) related_entity_key (Character) data_version_number (Integer)	None

Table 10–26 gives details of the table’s indexes.

Table 10–26: rym_data_version index information

Index name	Elements	Type
XPKrym_data_version	data_version_obj	Primary Unique
XAKIrym_data_version	related_entity_mnemonic related_entity_key	Unique

The update of this table should be automated by the version control procedures if they are being used to control maintenance of the data.

This information must be made available generically to a Help About window in the context of SmartObject versioning.

The version number for a piece of data is written by the versioning procedures. There is no method to generically handle versioning data when a user changes it outside of the version control procedures.

Security Group

This chapter covers the group of tables in the Progress Dynamics Repository that store information about application security. This chapter covers the following topics:

- [Security group structure](#)
- [Managers](#)
- [Table descriptions](#)

11.1 Security group structure

The Security group stores information about security allocations in your application. [Figure 11–1](#) shows the structure and relationships of the tables in this group.

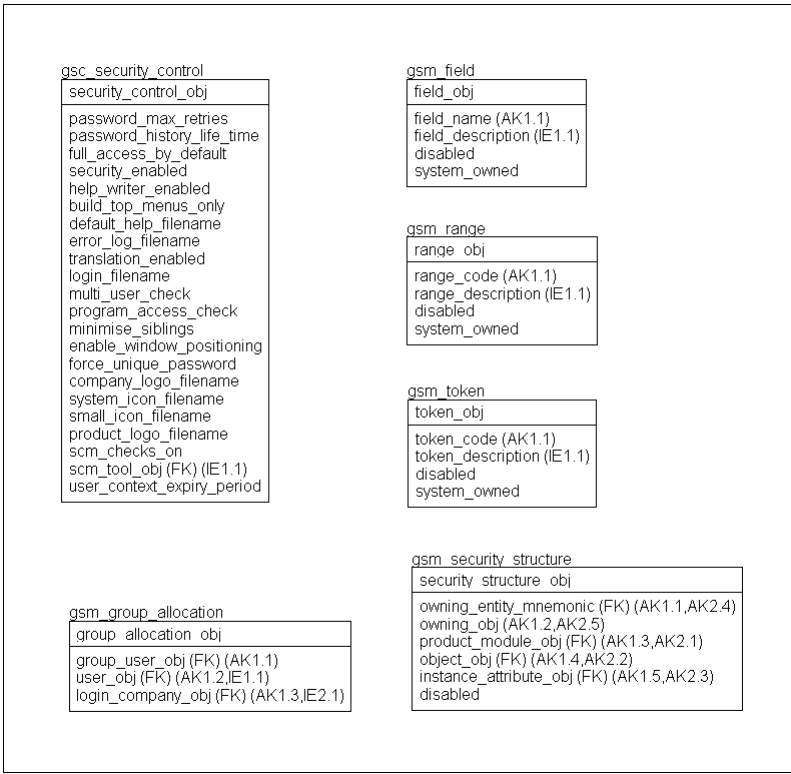


Figure 11–1: Security group

11.2 Managers

The Security Manager uses data from these tables to apply security allocations to your applications. For more detail about the temp-tables and APIs the Security Manager uses to manipulate data from the Security group tables, see the chapter on the Security Manager in the [Progress Dynamics Managers API Reference](#).

11.3 Table descriptions

The following sections provide detailed information about each table in the Security group. The Security group contains the following tables:

- [gsc_security_control table—gscsc](#)
- [gsm_field table—gsmff](#)
- [gsm_group_allocation table—gscsm](#)
- [gsm_range table—gsmra](#)
- [gsm_security_structure table—gsmss](#)
- [gsm_token table—gsmt0](#)

11.3.1 gsc_security_control table—gscsc

This table contains extra control information pertinent to security settings. This table only contains a single record.

Table 11–1 lists the table’s FLA, fields, and foreign keys.

Table 11–1: gsc_security_control table information

Table FLA	Fields (data type)	Foreign keys
gscsc	security_control_obj (Decimal) password_max_retries (Integer) password_history_life_time (Integer) full_access_by_default (Logical) security_enabled (Logical) help_writer_enabled (Logical) build_top_menus_only (Logical) default_help_filename (Character) error_log_filename (Character) translation_enabled (Logical) login_filename (Character) multi_user_check (Logical) program_access_check (Logical) minimise_siblings (Logical) enable_window_positioning (Logical) force_unique_password (Logical) company_logo_filename (Character) system_icon_filename (Character) small_icon_filename (Character) product_logo_filename (Character) scm_checks_on (Logical) scm_tool_obj (Decimal) user_context_expiry_period (Decimal)	scm_tool_obj

Table 11–2 gives details of the table’s indexes.

Table 11–2: gsc_security_control index information

Index name	Elements	Type
XPkgsc_security_control	security_control_obj	Primary Unique
XIE1gsc_security_control	scm_tool_obj	Nonunique

11.3.2 **gsm_field table—gsmff**

This table lists fields that require secured access in your application. Users can only receive restricted access to fields specified in this table. If a user is given restricted access to a field specified in this table, the access can be view only, hidden, or update.

[Table 11–3](#) lists the table’s FLA, fields, and foreign keys.

Table 11–3: gsm_field table information

Table FLA	Fields (data type)	Foreign keys
gsmff	field_obj (Decimal) field_name (Character) field_description (Character) disabled (Logical) system_owned (Logical)	field_name

[Table 11–4](#) gives details of the table’s indexes.

Table 11–4: gsm_field index information

Index name	Elements	Type
XPkgsm_field	field_obj	Primary Unique
XAK1gsm_field	field_name	Unique
XIE1gsm_field	field_description	Nonunique

For field security to be activated, entries must be created in the `gsm_security_structure` table. The `gsm_security_structure` table is allocated to users and allows the field security to be globally restricted or different in various parts of the application.

11.3.3 **gsm_group_allocation table—gscsm**

This table stores records of the groups to which a user belongs. This information is used for security purposes. You can also set up groups that belong to other groups. If the login company is specified, this indicates that security for the group only applies when logged in as the specified company.

[Table 11–5](#) lists the table’s FLA, fields, and foreign keys.

Table 11–5: gsm_group_allocation table information

Table FLA	Fields (data type)	Foreign keys
gsmga	group_allocation_obj (Decimal) group_user_obj (Decimal) user_obj (Decimal) login_company_obj (Decimal)	login_company_obj user_obj

[Table 11–6](#) gives details of the table’s indexes.

Table 11–6: gsm_group_allocation index information

Index name	Elements	Type
XPKgsm_group_allocation	group_allocation_obj	Primary Unique
XAK1gsm_group_allocation	group_user_obj user_obj login_company_obj	Unique
XIE1gsm_group_allocation	user_obj	Nonunique
XIE2gsm_group_allocation	login_company_obj	Nonunique

11.3.4 **gsm_range table—gsmra**

This table contains the definitions of ranges for your application's security controls.

[Table 11–7](#) lists the table's FLA, fields, and foreign keys.

Table 11–7: gsm_range table information

Table FLA	Fields (data type)	Foreign keys
gsmra	range_obj (Decimal) range_code (Character) range_description (Character) disabled (Logical) system_owned (Logical)	None

[Table 11–8](#) gives details of the table's indexes.

Table 11–8: gsm_range index information

Index name	Elements	Type
XPkgsm_range	range_obj	Primary Unique
XAKlgsm_range	range_code	Unique
XIElgsm_range	range_description	Nonunique

The range structures control what data specific users can view. When you allocate a range to a user, you are specifying what data the user is permitted to view. The appropriate data is hidden from the user.

Sample range structures might include “Nominal Codes,” “Cost Centers,” or “Member Codes.”

For range security to be activated, entries must be created in the `gsm_security_structure` table. The `gsm_security_structure` table is allocated to users and allows the range security to be globally restricted or different in various parts of the application.

11.3.5 **gsm_security_structure table—gsmss**

This table lists the parts of an application to which security restrictions apply. Currently access tokens, fields, and ranges are supported. The owning_obj refers either to a gsm_token record, a gsm_field record, or a gsm_range record.

Table 11–9 lists the table’s FLA, fields, and foreign keys.

Table 11–9: gsm_security_structure table information

Table FLA	Fields (data type)	Foreign keys
gsmss	security_structure_obj (Decimal) owning_entity_mnemonic (Character) owning_obj (Decimal) product_module_obj (Decimal) object_obj (Decimal) instance_attribute_obj (Decimal) disabled (Logical)	instance_attribute_obj product_module_obj

Table 11–10 gives details of the table’s indexes.

Table 11–10: gsm_security_structure index information

Index name	Elements	Type
XPKgsm_security_structure	security_structure_obj	Primary Unique
XAK1gsm_security_structure	owning_entity_mnemonic owning_obj product_module_obj object_obj instance_attribute_obj	Unique
XAK2gsm_security_structure	product_module_obj object_obj instance_attribute_obj owning_entity_mnemonic owning_obj	Unique

Because the fields are identical, one table is used rather than a usage table for each type of security allocation. If another type is introduced, no major rewrites are required because this table can automatically support the new type.

A security restriction can be assigned globally, in which case the product module, object, and instance attribute are 0. Alternately, the restriction can be allocated to a product module, a specific program object, or an instance attribute for a program. Because users are allocated the entries in this table, a restriction must be assigned to this table to be active.

11.3.6 **gsm_token table—gsmto**

This table contains the definitions of security tokens.

[Table 11–11](#) lists the table’s FLA, fields, and foreign keys.

Table 11–11: gsm_token table information

Table FLA	Fields (data type)	Foreign keys
gsmto	token_obj (Decimal) token_code (Character) token_description (Character) disabled (Logical) system_owned (Logical)	None

[Table 11–12](#) gives details of the table’s indexes.

Table 11–12: gsm_token index information

Index name	Elements	Type
XPkgsm_token	token_obj	Primary Unique
XAK1gsm_token	token_code	Unique
XIE1gsm_token	token_description	Nonunique

Tokens are used in the application to control access to functions the user can perform within a program using tab folder page names and button names.

Tokens can be created for any tab page names or button labels. Do not include shortcut characters and “...” suffixes. The tokens must be added to the gsm_security_structure table to become active. The gsm_security_structure table enables restricting a token for a specific object instance, specific object, specific product module, or generically for everything.

The framework only checks security if a valid, enabled token exists for the button label or tab folder page.

Once a user is allocated tokens, security restrictions are applied. If a folder page or button has access restrictions set up, only users who have been granted access can use it. However, if security control is set to full access by default and a user has no tokens allocated, then that user is assumed to have full access.

Example tokens would be add, delete, modify, view, copy, page 1, and page 2.

Sequence Group

This chapter covers the group of tables in the Progress Dynamics Repository that store information about sequences in your application. This chapter covers the following topics:

- [Sequence group structure](#)
- [Table descriptions](#)

12.1 Sequence group structure

The Sequence group stores information about the use of sequences specific to Progress Dynamics. [Figure 12–1](#) shows the structure and relationships of the tables in this group.

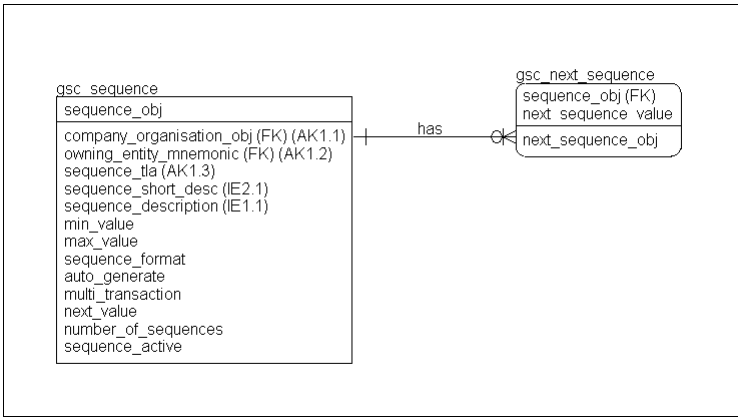


Figure 12–1: Sequence group

12.2 Table descriptions

The following sections provide detailed information about each table in the Sequence group. The Sequence group contains the following tables:

- [gsc_sequence table—gscsq](#)
- [gsc_next_sequence table—gscsn](#)

12.2.1 gsc_sequence table—gscsq

This is a generic sequence number and format table. All entries in this table are system-owned by their nature.

When a sequence number must be generated, it is done at the end of the update as part of the transaction. This table is a potential bottleneck, so locks should be kept to an absolute minimum. Locks during user interaction should be avoided.

If you want sequence numbers to be automatically generated, then there can be no holes in the sequence numbers. For this reason, a standard Progress sequence is not used.

Example uses for this table are the automatic generation of document numbers or transaction references.

Table 12–1 lists the table’s FLA, fields, and foreign keys.

Table 12–1: gsc_sequence table information

Table FLA	Fields (data type)	Foreign keys
gscsq	sequence_obj (Decimal) company_organisation_obj (Decimal) owning_entity_mnemonic (Character) sequence_tla (Character) sequence_short_desc (Character) sequence_description (Character) min_value (Integer) max_value (Integer) sequence_format (Character) auto_generate (Logical) multi_transaction (Logical) next_value (Integer) number_of_sequences (Integer) sequence_active (Logical)	sequence_obj

Table 12–2 gives details of the table’s indexes.

Table 12–2: gsc_sequence index information

Index name	Elements	Type
XPKgsc_sequence	sequence_obj	Primary Unique
XAK1gsc_sequence	company_organisation_obj owning_entity_mnemonic sequence_tla	Unique
XIE1gsc_sequence	sequence_description	Nonunique
XIE2gsc_sequence	sequence_short_desc	Nonunique

12.2.2 gsc_next_sequence table—gscsn

This table allocates sequence numbers where multiple sequence numbers might be requested by multiple transactions simultaneously. It is intended to avoid “deadly embrace” record locks.

Table 12–3 lists the table’s FLA, fields, and foreign keys.

Table 12–3: gsc_next_sequence table information

Table FLA	Fields (data type)	Foreign keys
gscsn	sequence_obj (Decimal) next_sequence_value (Integer) next_sequence_obj (Decimal)	sequence_obj

Table 12–4 gives details of the table’s indexes.

Table 12–4: gsc_next_sequence index information

Index name	Elements	Type
XPKgsc_next_sequence	sequence_obj next_sequence_value	Primary Unique

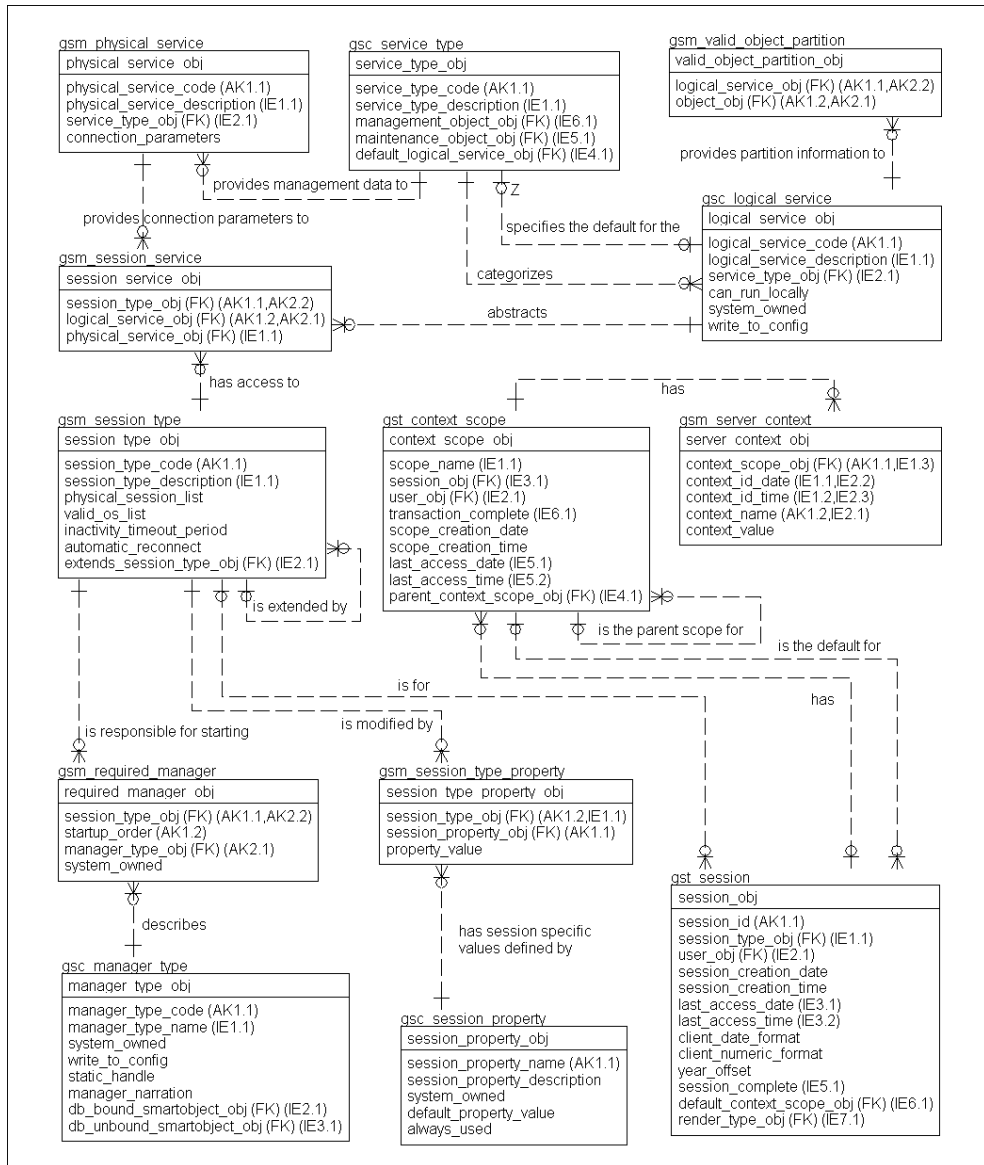
When a sequence is created or updated on the gsc_sequence table with the multi_transaction field set to YES, a number of sequence records equal to the number_of_sequences field’s value are created in this table. The new sequence numbers start with the value listed in the next_sequence_value field.

When a sequence number is requested, the first record in this table for the gsc_sequence is found, saved, and deleted. At the same time, a new gsc_next_sequence record is tagged on the end, with the sequence number just found plus the number_of_sequences value.

Session and Configuration Group

This chapter covers the group of tables in the Progress Dynamics Repository that store information about sessions and configurations. This chapter covers the following topics:

- [Session and Configuration group structure](#)
- [Managers](#)
- [Table descriptions](#)



13.2 Managers

The managers involved in starting up and managing the Progress Dynamics framework use the data on these tables. These managers include the Configuration File, Connection, General, Service Type, and Session Managers. For more detail about the temp-tables and APIs these managers use to manipulate data from the Session and Configuration group tables, see the chapters on these managers in the *Progress Dynamics Managers API Reference*.

13.3 Table descriptions

The following sections provide detailed information about each table in the Session and Configuration group. The Session and Configuration group contains the following tables:

- [gsm_session_type table—gscse](#)
- [gsc_logical_service table—gscls](#)
- [gsc_manager_type table—gscmt](#)
- [gsc_service_type table—gscst](#)
- [gsc_session_property table—gscsp](#)
- [gsm_physical_service table—gsmpy](#)
- [gsm_required_manager table—gsmrm](#)
- [gsm_server_context table—gsmsc](#)
- [gsm_session_service table—gsmsv](#)
- [gsm_session_type_property table—gsmsy](#)
- [gsm_valid_object_partition table—gsmvp](#)
- [gst_context_scope table—gstcs](#)
- [gst_session table—gstss](#)

13.3.1 **gsm_session_type table—gscse**

The gsc_session_type table is the central table of the Session and Configuration group. This table maintains the list of supported session types. This table also holds parameter details for the session, such as whether to support failover and inactivity time-outs for sessions.

Table 13–1 lists the table’s FLA, fields, and foreign keys.

Table 13–1: gsm_session_type table information

Table FLA	Fields (data type)	Foreign keys
gsmse	session_type_obj (Decimal) session_type_code (Character) session_type_description (Character) physical_session_list (Character) valid_os_list (Character) inactivity_timeout_period (Decimal) automatic_reconnect (Logical) extends_session_type_obj (Decimal)	session_type_obj

Table 13–2 gives details of the table’s indexes.

Table 13–2: gsm_session_type index information

Index name	Elements	Type
XPKgsm_session_type	session_type_obj	Primary Unique
XAK1gsm_session_type	session_type_code	Unique
XIE1gsm_session_type	session_type_description	Nonunique
XIE2gsm_session_type	extends_session_type_obj	Nonunique

A session type is a namespace for grouping specific types of sessions together. For example, some session types might be a “receipting workstation” and a “salesman’s Web agent.”

Each session type is combined with a physical session type that maps to a list of known Progress 4GL run-time environments. Thus, “receipting workstation” might map to a 4GL GUI client, and “salesman’s Web agent” might map to a WebSpeed® Transaction Agent. You can create any number of session types, mapping them to specific 4GL session types.

Where different managers need to be prestarted for different applications, the different applications are defined as new session types.

To run locally without Progress AppServer™ connections, you would define a new session type that connects to appropriate databases and has no session service records for the AppServer logical services. That would force them to not connect, using the session handle for code portability instead. This new session type would define a different set of managers to run as well. It would run the server-side managers locally.

There is a recursive join to the `gsc_session_type` table to support inheritance of settings across multiple session types. This makes the modification, reuse, and creation of session types easier. The program that generates the `icfconfig.xml` file reads the extra details from the linked session types and writes out an accumulated set of configuration details.

13.3.2 `gsc_logical_service` table—`gscls`

This table defines the logical services available to the application. A logical service is a separate process running either locally or remotely that requires connection parameters to establish communication between the client and the service. Logical service names are unique so that connection to the service can be completely abstracted from the developer.

The physical service determines the actual connection parameters. Which physical service is used is determined by combining the session type with the logical service.

Logical services for AppServer service types are AppServer partition names. Logical services for database connection service types are the logical database name.

[Table 13–3](#) lists the table’s FLA, fields, and foreign keys.

Table 13–3: `gsc_logical_service` table information

Table FLA	Fields (data type)	Foreign keys
gscls	logical_service_obj (Decimal) logical_service_code (Character) logical_service_description (Character) service_type_obj (Decimal) can_run_locally (Logical) system_owned (Logical) write_to_config (Logical)	logical_service_obj service_type_obj

Table 13–4 gives details of the table’s indexes.

Table 13–4: gsc_logical_service index information

Index name	Elements	Type
XPKgsc_logical_service	logical_service_obj	Primary Unique
XAK1gsc_logical_service	logical_service_code	Unique
XIE1gsc_logical_service	logical_service_description	Nonunique
XIE2gsc_logical_service	service_type_obj	Nonunique

13.3.3 gsc_manager_type table—gscmt

This table contains the definition of the standard Progress Dynamics managers.

Table 13–5 lists the table’s FLA, fields, and foreign keys.

Table 13–5: gsc_manager_type table information

Table FLA	Fields (data type)	Foreign keys
gscmt	manager_type_obj (Decimal) manager_type_code (Character) manager_type_name (Character) system_owned (Logical) write_to_config (Logical) static_handle (Character) manager_narration (Character) db_bound_smartobject_obj (Decimal) db_unbound_smartobject_obj (Decimal)	manager_type_obj

Table 13–6 gives details of the table’s indexes.

Table 13–6: gsc_manager_type index information

Index name	Elements	Type
XPKgsc_manager_type	manager_type_obj	Primary Unique
XAK1gsc_manager_type	manager_type_code	Unique
XIE1gsc_manager_type	manager_type_name	Nonunique
XIE2gsc_manager_type	db_bound_smartobject_obj	Nonunique
XIE3gsc_manager_type	db_unbound_smartobject_obj	Nonunique

A predefined set of managers that support the framework functionality exists in this table. Only managers that directly affect the core functionality of the Progress Dynamics framework need to be defined as manager types. For example, because no other code can be started without the Session Manager, the Session Manager must be listed.

The physical procedures to run for a manager type are defined in this table. Two standard physical procedures are supported:

- The bound manager object is for sessions that have a database connection service type as one of its session services. A manager of this type requires a database connection and communicates directly with schema tables.
- The unbound manager object is for sessions that do not have a database connection. A manager of this type does not require a database connection and communicates indirectly with schema tables through a bound manager object running on the server in a session with a database connection.

To support overrides of standard manager functionality, the super procedure attribute can be used against the manager object to define a super procedure stack of override functionality. It would point at a specific procedure to add as a super procedure, which in turn could have a super procedure, and so on.

The bound and unbound manager fields might need to be customized to point at the last procedure in the super procedure stack. To ensure that the customizations do not get lost, these fields must be defined as fields that are not overridden by new deployments. This is defined in the `exclude_fields` field of the `gsc_dataset_entity` table.

The manager objects specified here are always started for any session type that includes the manager type as a required manager type on the gsm_required_manager table. Because of this approach, different manager objects cannot be started for different session types. If this is required, then different manager types must be defined

13.3.4 gsc_service_type table—gscst

This table describes the different types of services available to applications and provides the management procedures for the different types of connections. Database services, AppServer services, and JMS partitions are examples of different service types. The Database and AppServer services are system-owned.

Table 13–7 lists the table’s FLA, fields, and foreign keys.

Table 13–7: gsc_service_type table information

Table FLA	Fields (data type)	Foreign keys
gscst	service_type_obj (Decimal) service_type_code (Character) service_type_description (Character) management_object_obj (Decimal) maintenance_object_obj (Decimal) default_logical_service_obj (Decimal)	service_type_obj

Table 13–8 gives details of the table’s indexes.

Table 13–8: gsc_service_type index information

Index name	Elements	Type
XPKgsc_service_type	service_type_obj	Primary Unique
XAK1gsc_service_type	service_type_code	Unique
XIE1gsc_service_type	service_type_description	Nonunique
XIE4gsc_service_type	default_logical_service_obj	Nonunique
XIE5gsc_service_type	maintenance_object_obj	Nonunique
XIE6gsc_service_type	management_object_obj	Nonunique

The maintenance object defines the datafield object used to maintain the physical connection parameter attribute on the physical service table. For example, if this is a database connection service type, then the datafield can allow the specification of -S, -N, and -H prompts independently. It can then put the result as a single field into the connection parameter.

The management object is the API procedure that is responsible for making the physical connections to the service. In the case of an AppServer partition, the default logical service could point at the default logical AppServer partition to use.

13.3.5 gsc_session_property table—gscsp

This table contains the list of valid properties that can be specified in the “properties” node of the Progress Dynamics configuration file, `icfconfig.xml`.

[Table 13–9](#) lists the table’s FLA, fields, and foreign keys.

Table 13–9: gsc_session_property table information

Table FLA	Fields (data type)	Foreign keys
gscsp	session_property_obj (Decimal) session_property_name (Character) session_property_description (Character) system_owned (Logical) default_property_value (Character) always_used (Logical)	session_property_obj

[Table 13–10](#) gives details of the table’s indexes.

Table 13–10: gsc_session_property index information

Index name	Elements	Type
XPkgsc_session_property	session_property_obj	Primary Unique
XAKlgsc_session_property	session_property_name	Unique

These property values can be set and retrieved using Session Manager APIs. This enables you to use them to alter the way that the session performs, depending on the session type.

13.3.6 **gsm_physical_service table—gsmphy**

This table maintains the list of supported physical service.

Table 13–11 lists the table’s FLA, fields, and foreign keys.

Table 13–11: gsm_physical_service table information

Table FLA	Fields (data type)	Foreign keys
gsmphy	physical_service_obj (Decimal) physical_service_code (Character) physical_service_description (Character) service_type_obj (Decimal) connection_parameters (Character)	physical_service_obj service_type_obj

Table 13–12 gives details of the table’s indexes.

Table 13–12: gsm_physical_service index information

Index name	Elements	Type
XPKgsm_physical_service	physical_service_obj	Primary Unique
XAK1gsm_physical_service	physical_service_code	Unique
XIE1gsm_physical_service	physical_service_description	Nonunique
XIE2gsm_physical_service	service_type_obj	Nonunique

This table provides the specific connection parameters that are required to connect a physical service to a session. A physical service is connected to a logical service and session type by the session service.

The maintenance object on the service type defines the datafield object used to maintain the physical connection parameters attribute. For example, in a database connection service type, the datafield might allow the specification of -S, -N, and -H prompts independently. It can then put the result as a single field into the connection parameters.

The management object on the service type is the API procedure that is responsible for making the physical connections to the service.

13.3.7 **gsm_required_manager table—gsmrm**

This table contains a list of the Progress Dynamics managers that must be started during the startup of the session and the order in which they must be started. Any manager types that need to be written to the configuration file must be started first.

Table 13–13 lists the table’s FLA, fields, and foreign keys.

Table 13–13: gsm_required_manager table information

Table FLA	Fields (data type)	Foreign keys
gsmrm	required_manager_obj (Decimal) session_type_obj (Decimal) startup_order (Integer) manager_type_obj (Decimal) object_obj (Decimal) system_owned (Logical)	manager_type_obj session_type_obj startup_order

Table 13–14 gives details of the table’s indexes.

Table 13–14: gsm_required_manager index information

Index name	Elements	Type
XPKgsm_required_manager	required_manager_obj	Primary Unique
XAK1gsm_required_manager	session_type_obj startup_order	Unique
XAK2gsm_required_manager	manager_type_obj session_type_obj	Unique

The write_to_config attribute of the manager causes this procedure name to be written to the icfconfig.xml file. This allows the framework to start it before the session makes a connection to the run-time Repository.

The Progress Dynamics framework supports a standard set of managers that are required. Specific applications might require the startup of additional managers for performance reasons. For example, a financial system might require a frequently referenced financial manager API to be prestarted.

13.3.8 **gsm_server_context table—gsmssc**

This table is a generic table to store context information between stateless AppServer connections. The context information might include user information, security information, SCM workspace and task information, or web page field values. This is a child table of the gsm_context_scope table. It can contain context data for a session or a user.

Table 13–15 lists the table’s FLA, fields, and foreign keys.

Table 13–15: gsm_server_context table information

Table FLA	Fields (data type)	Foreign keys
gsmssc	server_context_obj (Decimal) context_scope_obj (Decimal) context_id_date (Date) context_id_time (Integer) context_name (Character) context_value (Character)	context_scope_obj

Table 13–16 gives details of the table’s indexes.

Table 13–16: gsm_server_context index information

Index name	Elements	Type
XPKgsm_server_context	server_context_obj	Primary Unique
XAK1gsm_server_context	context_scope_obj context_name	Unique
XIE1gsm_server_context	context_id_date context_id_time context_scope_obj	Nonunique
XIE2gsm_server_context	context_name context_id_date context_id_time	Nonunique

13.3.9 **gsm_session_service table—gsmsv**

This table defines the different physical services needed to establish the logical service for each session type.

[Table 13–17](#) lists the table's FLA, fields, and foreign keys.

Table 13–17: gsm_session_service table information

Table FLA	Fields (data type)	Foreign keys
gsmsv	session_service_obj (Decimal) session_type_obj (Decimal) logical_service_obj (Decimal) physical_service_obj (Decimal)	logical_service_obj physical_service_obj session_type_obj

[Table 13–18](#) gives details of the table's indexes.

Table 13–18: gsm_session_service index information

Index name	Elements	Type
XPkgsm_session_service	session_service_obj	Primary Unique
XAK1gsm_session_service	session_type_obj logical_service_obj	Unique
XAK2gsm_session_service	logical_service_obj session_type_obj	Unique
XIE1gsm_session_service	physical_service_obj	Nonunique

For example, an AppServer might require a shared memory connection to the Repository. However, a client session would require a network connection to the Repository. The physical connection parameters are different for each of these services. The logical service identifies the database that needs to be connected, and the physical service describes the mechanism for the connection dependant on the session type.

If no session service record exists, then the logical service can only run locally.

13.3.10 **gsm_session_type_property table—gsmsy**

This table resolves the many-to-many relationship between the gsc_session_property and gsm_session_type tables.

Table 13–19 lists the table’s FLA, fields, and foreign keys.

Table 13–19: gsm_session_type_property table information

Table FLA	Fields (data type)	Foreign keys
gsmsy	session_type_property_obj (Decimal) session_type_obj (Decimal) session_property_obj (Decimal) property_value (Character)	session_property_obj session_type_obj

Table 13–20 gives details of the table’s indexes.

Table 13–20: gsm_session_type_property index information

Index name	Elements	Type
XPKgsm_session_type_property	session_type_property_obj	Primary Unique
XAK1gsm_session_type_property	session_property_obj session_type_obj	Unique
XIE1gsm_session_type_property	session_type_obj	Nonunique

If a record is found in this table for a property and a session type, the value specified in that record is written to the Progress Dynamics configuration file for the parameter.

If no record exists for a given property and session type, the always_used flag on the gsc_session_property table is checked. If the flag is on, the default value in the default_property_value field on the gsc_session_property table is used.

13.3.11 **gsm_valid_object_partition table—gsmvp**

This table defines a list of valid partitions in which a procedure can be run. A partition is a logical AppServer partition.

[Table 13–21](#) lists the table's FLA, fields, and foreign keys.

Table 13–21: gsm_valid_object_partition table information

Table FLA	Fields (data type)	Foreign keys
gsmvp	valid_object_partition_obj (Decimal) logical_service_obj (Decimal) object_obj (Decimal)	logical_service_obj

[Table 13–22](#) gives details of the table's indexes.

Table 13–22: gsm_valid_object_partition index information

Index name	Elements	Type
XPKgsm_valid_object_partition	valid_object_partition_obj	Primary Unique
XAK1gsm_valid_object_partitio	logical_service_obj object_obj	Unique
XAK2gsm_valid_object_partitio	object_obj logical_service_obj	Unique

The list only contains records when the object is restricted to certain partitions. When the object can be run on any partition, there are no records in this table.

The records in this table are only applicable to AppServer session types.

13.3.12 **gst_context_scope table—gstcs**

This table defines the scope of server context data, either by session or by user. If the context scope is for a user, then the context data in gsm_server_context persists across sessions and remains valid until deleted or until it expires. There is a flag on gsc_security_control that indicates when user context data should expire. This aids storing context for a user that can be re-used in new sessions, such as shopping cart information that can persist between web sessions. For user context scope, the session_obj is set to 0 and a valid user_obj must be specified.

If the context scope is for a session, then the context data in the `gsm_server_context` table is only valid for the duration of a single session. In this case, the `session_obj` is specified and the `user_obj` is 0. A name is given to the scope record to identify the scope. This is useful for context scoped to a user, and can be used through APIs to retrieve specific types of scope. Where the scope name is not required or specified, it is automatically set to the string value of the `context_scope_obj` to make it a unique number. The scope name is only unique for active context, that is, while `transaction_complete` is no. Once context is complete, the scope name is irrelevant and can be duplicated. This unique validation must be handled in code.

When dealing with transaction data, many context scope records could exist for a single session for a single transaction. When this is the case, the parent scope object ID identifies which scope records together form the complete transaction. The record with a parent of 0 is the top parent scope. For example, this happens on the WEB when dealing with parent and child data, such as order and order lines as a single transaction across multiple web pages. An order might contain many order lines, and the order lines have a common set of fields with different values in each case. The data stored in the `gsm_server_context` table points at a different context scope record to handle the same data fields existing in context for different records. The framework can gather context data across multiple web pages for multiple records, and then commit the entire set of data as a single transaction.

The `transaction_complete` flag is only set for the top parent context scope where the object ID is 0. It indicates that the transaction is finished and can be tidied up by the framework. The `session_complete` flag on `gst_session` can override this behavior when that flag is set to YES and the context data is session scoped. Additionally, the scope name must be unique for context scope where the `transaction_complete` flag is no.

Table 13–23 lists the table’s FLA, fields, and foreign keys.

Table 13–23: `gst_context_scope` table information

Table FLA	Fields (data type)	Foreign keys
gstcs	context_scope_obj (Decimal) scope_name (Character) session_obj (Decimal) user_obj (Decimal) transaction_complete (Logical) scope_creation_date (Date) scope_creation_time (Integer) last_access_date (Date) last_access_time (Integer) parent_context_scope_obj (Decimal)	context_scope_obj session_obj user_obj

Table 13–24 gives details of the table’s indexes.

Table 13–24: `gst_context_scope` index information

Index name	Elements	Type
XPKgst_context_scope	context_scope_obj	Primary Unique
XIE2gst_context_scope	user_obj scope_name transaction_complete	Nonunique
XIE3gst_context_scope	session_obj scope_name transaction_complete	Nonunique
XIE4gst_context_scope	parent_context_scope_obj	Nonunique

13.3.13 `gst_session` table—`gstss`

This table records a session and is only applicable for the duration of a session. It records session activity and manages the context for the session in the child table, `gsm_server_context`.

Table 13–25 lists the table’s FLA, fields, and foreign keys.

Table 13–25: `gst_session` table information

Table FLA	Fields (data type)	Foreign keys
gstss	session_obj (Decimal) session_id (Character) session_type_obj (Decimal) user_obj (Decimal) session_creation_date (Date) session_creation_time (Integer) last_access_date (Date) last_access_time (Integer) client_date_format (Character) client_numeric_format (Character) year_offset (Integer) session_complete (Logical) default_context_scope_obj (Decimal) render_type_obj (Decimal)	render_type_obj session_obj session_type_obj user_obj

[Table 13–26](#) gives details of the table’s indexes.

Table 13–26: gst_session index information

Index name	Elements	Type
XPkgst_session	session_obj	Primary Unique
XAK1gst_session	session_id	Unique
XIE1gst_session	session_type_obj	Nonunique
XIE2gst_session	user_obj	Nonunique
XIE3gst_session	last_access_date last_access_time	Nonunique
XIE5gst_session	session_complete	Nonunique
XIE6gst_session	default_context_scope_obj	Nonunique
XIE7gst_session	render_type_obj	Nonunique

The information in this table is all client related. In the case of a WebSpeed Agent, the agent is also the client.

The session id field from gsm_server_context table is now on this table, so the session id only exists in one place for the session. The object id for the session record is carried down onto the individual context records. This improves the efficiency of fixing the context id when a session is dropped and reconnected. That restores the context for the new session from the old session. For the AppServer, a record is created in this table at connection time and checked or updated in the activate procedure. To ensure that it updates as efficiently as possible, this table must not have a write trigger.

Status Group

This chapter covers the group of tables in the Progress Dynamics Repository that store information about categories and their status. This chapter covers the following topics:

- [Status group structure](#)
- [Table descriptions](#)

14.1 Status group structure

The Status group stores information about categories and their use in the framework and your applications. [Figure 14–1](#) shows the structure and relationships of the tables in this group.

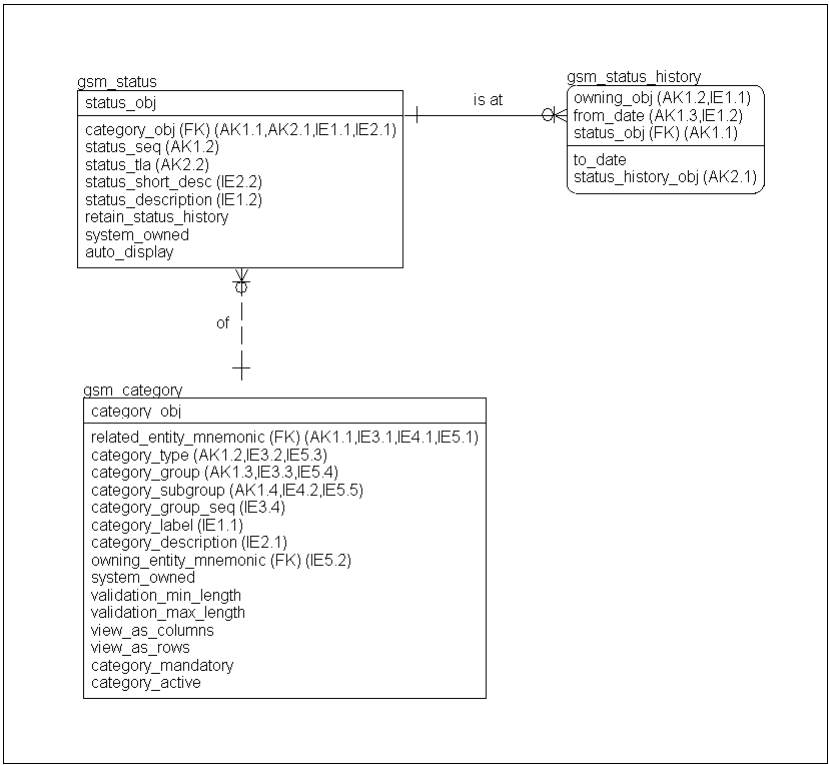


Figure 14–1: Status group

14.2 Table descriptions

The following sections provide detailed information about each table in the Status group. The Status group contains the following tables:

- [gsm_category table—gsmca](#)
- [gsm_status table—gsmst](#)
- [gsm_status_history table—gsmsh](#)

14.2.1 **gsm_category table—gsmca**

This table is a multi-purpose grouping mechanism for generic entities. Certain categories might be system-owned or generated and cannot be deleted. These categories are usually hard-coded into your application.

Some categories might not be associated with another generic entity. These are used to store hard-coded valid value lists, lookup lists, and similar information. Whenever hard-coded mnemonics are used in the Progress Dynamics framework, their usage and description is defined in this table.

[Table 14–1](#) lists the table’s FLA, fields, and foreign keys.

Table 14–1: gsm_category table information

Table FLA	Fields (data type)	Foreign keys
gsmca	category_obj (Decimal) related_entity_mnemonic (Character) category_type (Character) category_group (Character) category_subgroup (Character) category_group_seq (Integer) category_label (Character) category_description (Character) owning_entity_mnemonic (Character) system_owned (Logical) validation_min_length (Integer) validation_max_length (Integer) view_as_columns (Integer) view_as_rows (Integer) category_mandatory (Logical) category_active (Logical)	category_obj

Table 14–2 gives details of the table’s indexes.

Table 14–2: gsm_category index information

Index name	Elements	Type
XPKgsm_category	category_obj	Primary Unique
XAK1gsm_category	related_entity_mnemonic category_type category_group category_subgroup	Unique
XIE1gsm_category	category_label	Nonunique
XIE2gsm_category	category_description	Nonunique
XIE3gsm_category	related_entity_mnemonic category_type category_group category_group_seq	Nonunique
XIE4gsm_category	related_entity_mnemonic category_subgroup	Nonunique
XIE5gsm_category	related_entity_mnemonic owning_entity_mnemonic category_type category_group category_subgroup	Nonunique

14.2.2 gsm_status table—gsmst

This table lists the status of categories. You can use this table to modify the narrative of a status and to add extra status levels within a category subgroup to represent your internal business processes.

The valid status codes are set up in the gsm_category table, for example:

- Related entity mnemonic – GSMST for gsm_status
- Category type – STS for Status
- Category group – HST for History
- Category subgroup – COD for Code

Table 14–3 lists the table’s FLA, fields, and foreign keys.

Table 14–3: gsm_status table information

Table FLA	Fields (data type)	Foreign keys
gsmst	status_obj (Decimal) category_obj (Decimal) status_seq (Integer) status_tla (Character) status_short_desc (Character) status_description (Character) retain_status_history (Logical) system_owned (Logical) auto_display (Logical)	category_obj status_obj

Table 14–4 gives details of the table’s indexes.

Table 14–4: gsm_status index information

Index name	Elements	Type
XPkgsm_status	status_obj	Primary Unique
XAK1gsm_status	category_obj status_seq	Unique
XAK2gsm_status	category_obj status_tla	Unique
XIE1gsm_status	category_obj status_description	Nonunique
XIE2gsm_status	category_obj status_short_desc	Nonunique

The category subgroup is the actual status in this case. The status categories are always system-owned and mandatory. The category mandatory flag indicates whether or not an object at this status can be modified.

At least one record must exist in this table for every category subgroup that has a related entity mnemonic of status (GSMST). The record is system-owned, has a sequence of 0, and cannot be deleted. It is always the default status for this category subgroup.

From a business logic point of view, when a status changes within the same category, nothing needs to be done. The user can do this manually by a combo box. Changing status from one category to the next within a category group is usually done by a business logic process.

A join back to the `gsm_category` table determines what status an object is from a business logic point of view.

Entries on the `gsm_status_history` table determine the effective status of objects from specific dates. Therefore, the status does not need to be added to every table where it is used. For performance reasons in some cases, objects are linked directly to the `gsm_status` table to show the current status.

14.2.3 `gsm_status_history` table—`gsmsh`

This table lists the actual status of an object from its effective date.

[Table 14–5](#) lists the table’s FLA, fields, and foreign keys.

Table 14–5: `gsm_status_history` table information

Table FLA	Fields (data type)	Foreign keys
gsmsh	owning_obj (Decimal) from_date (Date) status_obj (Decimal) to_date (Date) status_history_obj (Decimal)	status_obj

Table 14–6 gives details of the table’s indexes.

Table 14–6: gsm_status_history index information

Index name	Elements	Type
XPKgsm_status_history	owning_obj from_date status_obj	Primary Unique
XAK1gsm_status_history	status_obj owning_obj from_date	Unique
XAK2gsm_status_history	status_history_obj	Unique
XIE1gsm_status_history	owning_obj from_date	Nonunique

Treeview Group

This chapter covers the group of tables in the Progress Dynamics Repository that store information about treeviews. This chapter covers the following topics:

- [Treeview group structure](#)
- [Managers](#)
- [Table descriptions](#)

15.1 Treeview group structure

The Treeview group stores information about the structure of treeviews used in your applications. [Figure 15–1](#) shows the structure and relationships of the tables in this group.

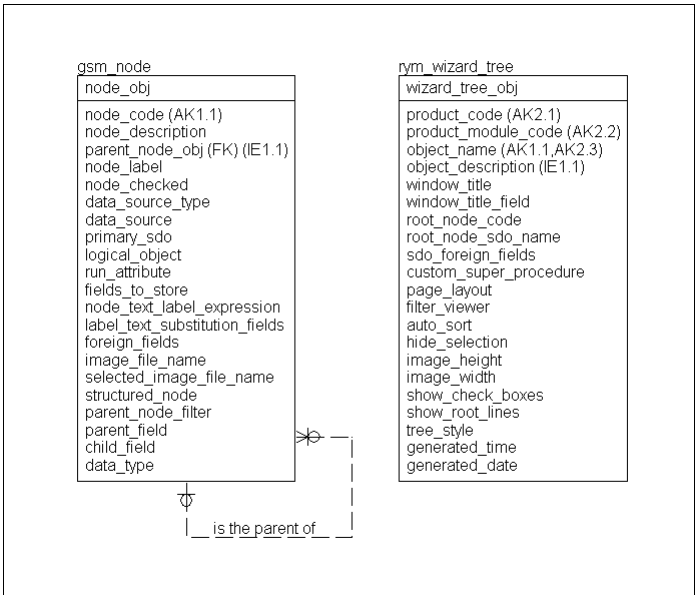


Figure 15–1: Treeview group

15.2 Managers

The Repository Managers use these tables to provide the Progress Dynamics framework with information to build application user interfaces. For more detail about the temp-tables and APIs the Repository Managers use to manipulate data from the Menu and Toolbar group tables, see the chapter on the Repository Managers in the [Progress Dynamics Managers API Reference](#).

15.3 Table descriptions

The following sections provide detailed information about each table in the Treeview group. The Treeview group contains the following tables:

- [gsm_node table—gsmnd](#)
- [rym_wizard_tree table—rymwt](#)

15.3.1 **gsm_node table—gsmnd**

This table records the parent-child relationships of node behavior for the TreeView controller.

This table includes support for structured nodes. Structured node are nodes where each new level is created infinitely from the same SDO. This means that a node can be expanded infinite times without having to set up a node for each level in node control.

For example, you might set the following field values on this table to create a structured node treeview:

```
structured_node - YES
parent_node_filter - parent_node_obj = 0
parent_field - parent_node_obj
child_field - node_obj
datatype - DECIMAL
```

[Table 15–1](#) lists the table’s FLA, fields, and foreign keys.

Table 15–1: gsm_node table information

Table FLA	Fields (data type)	Foreign keys
gsmnd	node_obj (Decimal) node_code (Character) node_description (Character) parent_node_obj (Decimal) node_label (Character) node_checked (Logical) data_source_type (Character) data_source (Character) primary_sdo (Character) logical_object (Character) run_attribute (Character) fields_to_store (Character) node_text_label_expression (Character) label_text_substitution_fields (Character) foreign_fields (Character) image_file_name (Character) selected_image_file_name (Character) structured_node (Logical) parent_node_filter (Character) parent_field (Character) child_field (Character) data_type (Character)	None

Table 15–2 gives details of the table’s indexes.

Table 15–2: gsm_node index information

Index name	Elements	Type
XPKgsm_node	node_obj	Primary Unique
XAKlgsm_node	node_code	Unique
XIElgsm_node	parent_node_obj	Nonunique

15.3.2 rym_wizard_tree table—rymwt

This table captures wizard responses for the creation or modification of the Progress Dynamics’ standard Dynamic TreeView Controller object.

The wizard responses are used to forward engineer the object into the Repository, generating all appropriate SmartObject instances and attributes. To simplify the data that must be captured, assumptions are made regarding the look and feel of a standard Dynamic TreeView Controller.

More complex specific modifications to an object can be made using the standard Repository Maintenance options.

This table also aids in the generation of the object for different user interfaces, such as Java.

Table 15–3 lists the table’s FLA, fields, and foreign keys.

Table 15–3: rym_wizard_tree table information

Table FLA	Fields (data type)	Foreign keys
rymwt	wizard_tree_obj (Decimal) product_code (Character) product_module_code (Character) object_name (Character) object_description (Character) window_title (Character) window_title_field (Character) root_node_code (Character) root_node_sdo_name (Character) sdo_foreign_fields (Character) custom_super_procedure (Character) page_layout (Character) filter_viewer (Character) auto_sort (Logical) hide_selection (Logical) image_height (Integer) image_width (Integer) show_check_boxes (Logical) show_root_lines (Logical) tree_style (Integer) generated_time (Integer) generated_date (Date)	product_code product_module_code

Table 15–4 gives details of the table’s indexes.

Table 15–4: rym_wizard_tree index information

Index name	Elements	Type
XPKrym_wizard_tree	wizard_tree_obj	Primary Unique
XAK1rym_wizard_tree	object_name	Unique
XAK2rym_wizard_tree	product_code product_module_code object_name	Unique
XIE1rym_wizard_tree	object_description	Nonunique

User Group

This chapter covers the group of tables in the Progress Dynamics Repository that store information about users. This chapter covers the following topics:

- [User group structure](#)
- [Managers](#)
- [Table descriptions](#)

16.1 User group structure

The User group stores information about users of your application. [Figure 16–1](#) shows the structure and relationships of the tables in this group.

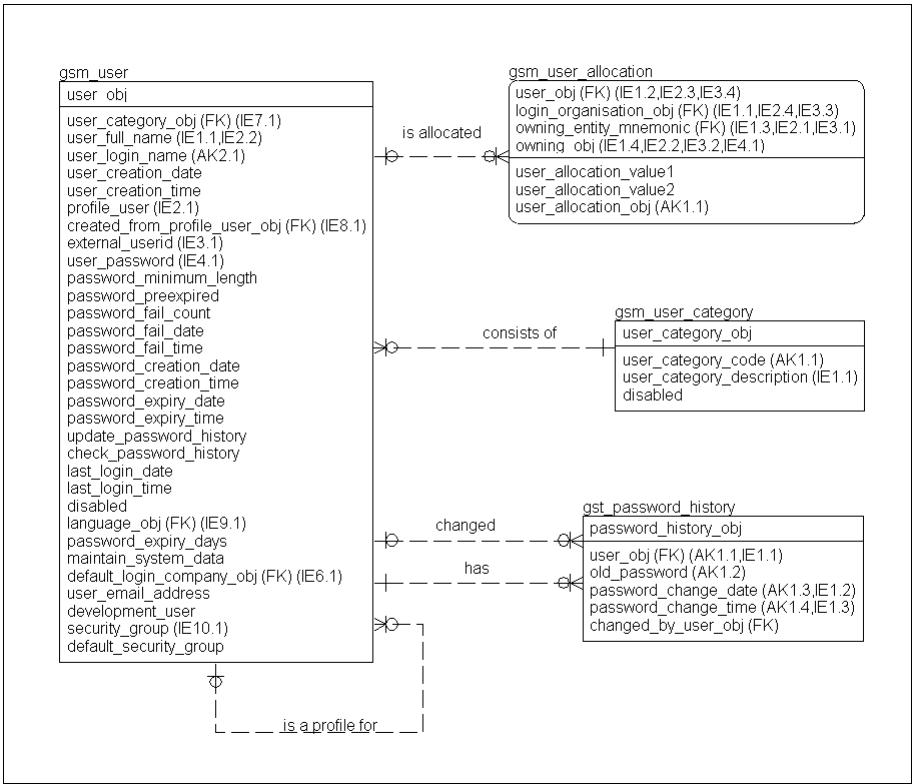


Figure 16–1: User group

16.2 Managers

Most of the Progress Dynamics Managers make use of information from the User group. User information is used to apply security allocations and user interface customizations. These tables are referenced throughout the framework to control what each user sees as they work with an application.

16.3 Table descriptions

The following sections provide detailed information about each table in the User group. The User group contains the following tables:

- [gsm_user table—gsmus](#)
- [gsm_user_allocation table—gsmul](#)
- [gsm_user_category table—gsmuc](#)
- [gst_password_history table—gstph](#)

16.3.1 **gsm_user table—gsmus**

The gsm_user table is the central table of the User group. This table defines the users who can log into the system, in other words, the approved users of the system.

Table 16–1 lists the table’s FLA, fields, and foreign keys.

Table 16–1: gsm_user table information

Table FLA	Fields (data type)	Foreign keys
gsmus	user_obj (Decimal) user_category_obj (Decimal) user_full_name (Character) user_login_name (Character) user_creation_date (Date) user_creation_time (Integer) profile_user (Logical) created_from_profile_user_obj (Decimal) external_userid (Integer) user_password (Character) password_minimum_length (Integer) password_preexpired (Logical) password_fail_count (Integer) password_fail_date (Date) password_fail_time (Integer) password_creation_date (Date) password_creation_time (Integer) password_expiry_date (Date) password_expiry_time (Integer) update_password_history (Logical) check_password_history (Logical) last_login_date (Date) last_login_time (Integer) disabled (Logical) language_obj (Decimal) password_expiry_days (Integer) maintain_system_data (Logical) default_login_company_obj (Decimal) user_email_address (Character) development_user (Logical) security_group (Logical) default_security_group (Logical)	language_obj user_category_obj user_obj

Table 16–2 gives details of the table’s indexes.

Table 16–2: gsm_user index information

Index name	Elements	Type
XPkgsm_user	user_obj	Primary Unique
XAK2gsm_user	user_login_name	Unique
XIE1gsm_user	user_full_name	Nonunique
XIE2gsm_user	profile_user user_full_name	Nonunique
XIE3gsm_user	external_userid	Nonunique
XIE4gsm_user	user_password	Nonunique
XIE6gsm_user	default_login_company_obj	Nonunique
XIE7gsm_user	user_category_obj	Nonunique
XIE8gsm_user	created_from_profile_user_ob	Nonunique
XIE9gsm_user	language_obj	Nonunique
XIE10gsm_user	security_group	Nonunique

The main user details are contained in an external security system, pointed at by the external_userid field. This table defines extra user information for a system, and allows a user to be optionally associated with a person to enable the entering of other data, such as full name, address, and comments.

There is a logged-in flag on the user record to enable the identification of user availability. Users are available if they are logged into the application.

The existence of this specific user table in the Repository also aids automatic referential integrity.

16.3.2 **gsm_user_allocation table—gsmul**

This table lists the security allocations for users and companies.

When users log into the system, they log in with a user ID and select a company (organization). This table defines the security options for users when they log into a certain organization. A user can have different security options when logged into different companies.

If the organisation_obj is 0, then the security allocation applies to all companies. Likewise, if the user_obj is 0, then the security allocation applies to all users logged into a company. User security always overrides company security.

NOTE: In addition, an owning_obj of 0 always indicates no access to any data for that entity mnemonic. This is not supported for security structures or menus. It is only for data.

This table generically assigns all security options for the user or company. The standard options that can be specified by the owning_entity_mnemonic and owning_obj are:

- gsm_security_structure records for security relating to tokens, fields, and ranges
- gsm_menu_items for securing access to menu items
- gsm_menu_structure for securing access to menu structures
- gsm_entity_field_value for securing access to generic entity field values, such as companies

Table 16–3 lists the table’s FLA, fields, and foreign keys.

Table 16–3: gsm_user_allocation table information

Table FLA	Fields (data type)	Foreign keys
gsmul	user_obj (Decimal) login_organisation_obj (Decimal) owning_entity_mnemonic (Character) owning_obj (Decimal) user_allocation_value1 (Character) user_allocation_value2 (Character) user_allocation_obj (Decimal)	user_obj

Table 16–4 gives details of the table’s indexes.

Table 16–4: gsm_user_allocation index information

Index name	Elements	Type
XPKgsm_user_allocation	user_obj login_organisation_obj owning_entity_mnemonic owning_obj	Primary Unique
XAK1gsm_user_allocation	user_allocation_obj	Unique
XIE1gsm_user_allocation	login_organisation_obj user_obj owning_entity_mnemonic owning_obj	Nonunique
XIE2gsm_user_allocation	owning_entity_mnemonic owning_obj user_obj login_organisation_obj	Nonunique
XIE3gsm_user_allocation	owning_entity_mnemonic owning_obj login_organisation_obj user_obj	Nonunique
XIE4gsm_user_allocation	owning_obj	Nonunique

Access to any entity data can be secured using this table. For example, to secure access to specific cost center codes in a general ledger, the owning_entity_mnemonic could be the cost center table and the owning_obj used to allocate the specific cost centers to which the user or company has access.

The rules applied to this table for the entity, in order, are as follows:

1. If security is disabled, then user security is passed.
2. If a specific record exists for the user or company, then security is passed.
3. If a specific record is found for the user or company with an owning_obj of 0, security is failed.
4. If full access is not granted by default, and there are no entries for the user, and there are no generic entries for all users and all companies, security is failed.

- 5. If a record exists for all users or all companies with an owning_obj of 0, then security is failed.
- 6. If a record is found for all users, security is passed.
- 7. If a record is found for all companies, security is passed.
- 8. If full access is granted by default and no records are found for the specific user, all users, or all companies, then security is passed.

Some allocations require additional data. For example, allocating a field restriction needs to determine what can be done with the field, such as View, Update, or Hide.

Entries must exist in this table for all security allocations. There is no option for inclusion or exclusion to make querying as fast as possible. The maintenance programs however should allow the specification by inclusion or exclusion for fast data entry, then create or delete all relevant entries in this table.

16.3.3 gsm_user_category table—gsmuc

This table defines categories of users. It could be used for job functions and similar categories. Its primary use is for filtering and reporting.

[Table 16–5](#) lists the table’s FLA, fields, and foreign keys.

Table 16–5: gsm_user_category table information

Table FLA	Fields (data type)	Foreign keys
gsmuc	user_category_obj (Decimal) user_category_code (Character) user_category_description (Character) disabled (Logical)	user_category_obj

Table 16–6 gives details of the table’s indexes.

Table 16–6: gsm_user_category index information

Index name	Elements	Type
XPKgsm_user_category	user_category_obj	Primary Unique
XAKlgsm_user_category	user_category_code	Unique
XIElgsm_user_category	user_category_description	Nonunique

16.3.4 gst_password_history table—gstph

This table keeps a history of passwords used previously by users. It is used for audit purposes and to prevent users from choosing the same password within a specified time period.

Table 16–7 lists the table’s FLA, fields, and foreign keys.

Table 16–7: gst_password_history table information

Table FLA	Fields (data type)	Foreign keys
gstph	password_history_obj (Decimal) user_obj (Decimal) old_password (Character) password_change_date (Date) password_change_time (Integer) changed_by_user_obj (Decimal)	user_obj

Table 16–8 gives details of the table’s indexes.

Table 16–8: gst_password_history index information

Index name	Elements	Type
XPKgst_password_history	password_history_obj	Primary Unique
XAK1gst_password_history	user_obj old_password password_change_date password_change_time	Unique
XIE1gst_password_history	user_obj password_change_date password_change_time	Nonunique

User Profile Group

This chapter covers the group of tables in the Progress Dynamics Repository that store information about user profiles. This chapter covers the following topics:

- [User Profile group structure](#)
- [Table descriptions](#)

17.1 User Profile group structure

The User Profile group stores information about user profiles. [Figure 17–1](#) shows the structure and relationships of the tables in this group.

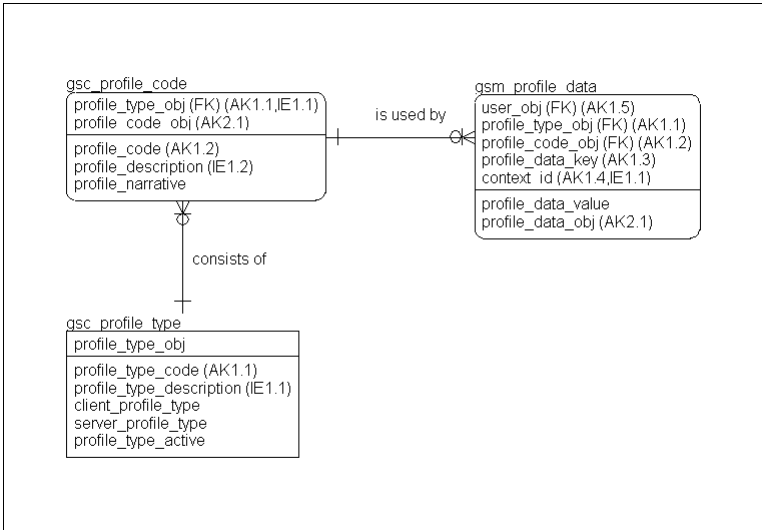


Figure 17–1: User Profile group

17.2 Table descriptions

The following sections provide detailed information about each table in the User Profile group. The User Profile group contains the following tables:

- [gsc_profile_code table—gscpc](#)
- [gsc_profile_type table—gscpf](#)
- [gsm_profile_data table—gsmpf](#)

17.2.1 gsc_profile_code table—gscpc

This table defines the codes that exist for each profile type. It defines the structure of the data key and data value fields when the code is allocated against a user in the gsm_profile_data table.

[Table 17–1](#) lists the table’s FLA, fields, and foreign keys.

Table 17–1: gsc_profile_code table information

Table FLA	Fields (data type)	Foreign keys
gscpc	profile_type_obj (Decimal) profile_code_obj (Decimal) profile_code (Character) profile_description (Character) profile_narrative (Character)	profile_type_obj

[Table 17–2](#) gives details of the table’s indexes.

Table 17–2: gsc_profile_code index information

Index name	Elements	Type
XPKgsc_profile_code	profile_type_obj profile_code_obj	Primary Unique
XAK1gsc_profile_code	profile_type_obj profile_code	Unique
XAK2gsc_profile_code	profile_code_obj	Unique
XIE1gsc_profile_code	profile_type_obj profile_description	Nonunique

An example of a profile type is filter settings. Examples of profile codes for filter settings are filter-from values, filter-to values, filtering enabled, and filter field names.

17.2.2 gsc_profile_type table—gscpf

This table defines the types of profile codes supported for allocation to users.

Table 17–3 lists the table’s FLA, fields, and foreign keys.

Table 17–3: gsc_profile_type table information

Table FLA	Fields (data type)	Foreign keys
gscpf	profile_type_obj (Decimal) profile_type_code (Character) profile_type_description (Character) client_profile_type (Logical) server_profile_type (Logical) profile_type_active (Logical)	profile_type_obj

Table 17–4 gives details of the table’s indexes.

Table 17–4: gsc_profile_type index information

Index name	Elements	Type
XPKgsc_profile_type	profile_type_obj	Primary Unique
XAK1gsc_profile_type	profile_type_code	Unique
XIE1gsc_profile_type	profile_type_description	Nonunique

Records exist for any type of profile information stored against a user between sessions. Examples include browser filter settings, report filter settings, toolbar customization settings, window positions and sizes, and systemwide settings like “tooltips on/off.”

17.2.3 gsm_profile_data table—gsmpf

This table is used to store profile information for specific users, such as browser filter settings, window positions and sizes, and report filter settings.

The nature of the profile_data_key and profile_data_value fields is determined by the profile type and code.

Data can be stored permanently, or for the current session only, depending on the context_id field.

Table 17–5 lists the table’s FLA, fields, and foreign keys.

Table 17–5: gsm_profile_data table information

Table FLA	Fields (data type)	Foreign keys
gsmprf	user_obj (Decimal) profile_type_obj (Decimal) profile_code_obj (Decimal) profile_data_key (Character) context_id (Character) profile_data_value (Character) profile_data_obj (Decimal)	profile_type_obj user_obj

Table 17–6 gives details of the table’s indexes.

Table 17–6: gsm_profile_data index information

Index name	Elements	Type
XPkgsm_profile_data	user_obj profile_type_obj profile_code_obj profile_data_key context_id	Primary Unique
XAK1gsm_profile_data	profile_type_obj profile_code_obj profile_data_key context_id user_obj	Unique
XAK2gsm_profile_data	profile_data_obj	Unique
XIE1gsm_profile_data	context_id	Nonunique

Other Tables

This chapter covers other tables in the Progress Dynamics Repository that are not part of a larger logical group. This chapter covers the following topics:

- [gst_audit table—gstad](#)
- [gsm_help table—gsmhe](#)
- [gsm_login_company table—gsmlg](#)

18.1 **gst_audit table—gstad**

This table stores information about auditing your application. [Figure 18–1](#) shows the structure of the table.

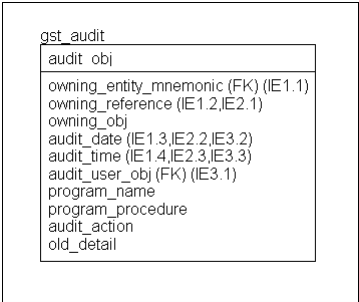


Figure 18–1: **gst_audit table**

18.1.1 Table description

This table acts as a global audit file to record modifications to data. The audit can be turned on by defining a category of audit for an entity type. It can be turned off again by resetting the active flag on a category.

[Table 18–1](#) lists the table’s FLA, fields, and foreign keys.

Table 18–1: **gst_audit table information**

Table FLA	Fields (data type)	Foreign keys
gstad	audit_obj (Decimal) owning_entity_mnemonic (Character) owning_reference (Character) owning_obj (Decimal) audit_date (Date) audit_time (Integer) audit_user_obj (Decimal) program_name (Character) program_procedure (Character) audit_action (Character) old_detail (Character)	None

Table 18–2 gives details of the table’s indexes.

Table 18–2: gst_audit index information

Index name	Elements	Type
XPkgst_audit	audit_obj	Primary Unique
XIE1gst_audit	owning_entity_mnemonic owning_reference audit_date audit_time	Nonunique
XIE2gst_audit	owning_reference audit_date audit_time	Nonunique
XIE3gst_audit	audit_user_obj audit_date audit_time	Nonunique

The audit holds the following information:

- Basic details on the action (create, amend, or delete)
- The user who performed the action
- The date and time of the action
- The program and procedure that performed the action

The audit might also hold a record of the data before the update. The audit can be used to keep old values of fields by defining more categories, such as one for each field or group of fields on an entity.

18.2 gsm_help table—gsmhe

This table stores information about application help. [Figure 18–2](#) shows the structure of the table.

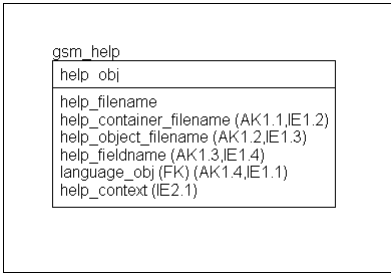


Figure 18–2: gsm_help table

18.2.1 Table description

This table defines help contexts for containers, objects on containers, and fields on objects if required.

When context-sensitive help is requested, the context and help file, if available, are retrieved from this file. If necessary, help can be specified in multiple languages.

An entry in this table for a specific language that does not specify a container, object, or field overrides the standard help file used systemwide for the specified language from the gsc_security_control table.

[Table 18–3](#) lists the table’s FLA, fields, and foreign keys.

Table 18–3: gsm_help table information

Table FLA	Fields (data type)	Foreign keys
gsmhe	help_obj (Decimal) help_filename (Character) help_container_filename (Character) help_object_filename (Character) help_fieldname (Character) language_obj (Decimal) help_context (Character)	language_obj

Table 18–4 gives details of the table’s indexes.

Table 18–4: gsm_help index information

Index name	Elements	Type
XPkgsm_help	help_obj	Primary Unique
XAK1gsm_help	help_container_filename help_object_filename help_fieldname language_obj	Unique
XIE1gsm_help	language_obj help_container_filename help_object_filename help_fieldname	Nonunique
XIE2gsm_help	help_context	Nonunique

18.3 gsm_login_company table—gsmlog

This table stores the definitions of login companies for your application. Figure 18–3 shows the structure of the table.

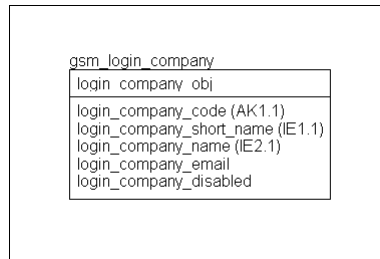


Figure 18–3: gsm_login_company table

18.3.1 Table description

This table defines login companies (organizations) and provides a list of valid companies the user can log into at run time.

You can link this table into an external application database where additional details are stored for the login company, such as address and contact information. The code or the object field could be used as the cross-reference into the external application database.

The Progress Dynamics framework uses this table in the generic setup of framework data specific to login companies, such as security allocations and automatic reference numbers.

The existence of a login company in the framework supports the concept of holding application data for multiple companies in a single database, instead of having separate databases for each company. Application databases have to link to this table, or a corresponding table in their database design, to filter appropriate data for each login company.

Table 18–5 lists the table’s FLA, fields, and foreign keys.

Table 18–5: gsm_login_company table information

Table FLA	Fields (data type)	Foreign keys
gsm1g	login_company_obj (Decimal) login_company_code (Character) login_company_short_name (Character) login_company_name (Character) login_company_email (Character) login_company_disabled (Logical)	login_company_obj

Table 18–6 gives details of the table’s indexes.

Table 18–6: gsm_login_company index information

Index name	Elements	Type
XPKgsm_login_company	login_company_obj	Primary Unique
XAK1gsm_login_company	login_company_code	Unique
XIE1gsm_login_company	login_company_short_name	Nonunique
XIE2gsm_login_company	login_company_name	Nonunique

This table only holds the minimum details required by the framework. In order to simplify and synchronize maintenance of this data, applications that also have an organization table should replicate modifications from their full organization table into this table.

Legacy and Developing Structures

This chapter covers groups of tables that are included in the Progress Dynamics Repository to support legacy functionality or to aid possible developments in the Progress Dynamics framework. This chapter covers the following topics:

- [Custom Procedure group](#)
- [Flow group](#)
- [Profile group](#)
- [Reporting Group](#)

19.1 Custom Procedure group

This section covers the group of tables in the Progress Dynamics Repository that store information about custom procedures. [Figure 19–1](#) shows the structure and relationships of the tables in this group.

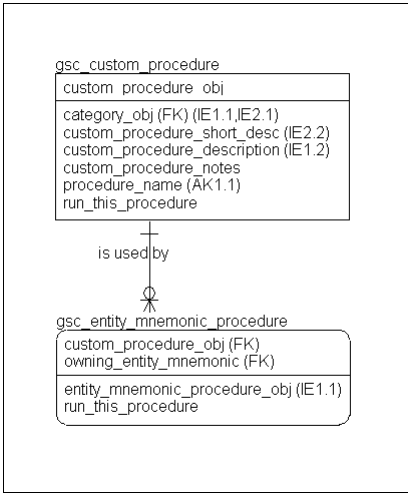


Figure 19–1: Custom Procedure group

The Custom Procedure group is in the Progress Dynamics Repository to support legacy applications built on the Astra framework. It supports setting up business logic that needs to change depending on user variables. For example, a sales application that serves users in several regions might need to calculate sales tax differently based on region.

The Flow group is a newer, developing approach to this situation.

The Custom Procedure group contains the following tables:

- `gsc_custom_procedure` table
- `gsc_entity_mnemonic_procedure` table

19.2 Flow group

This section covers the group of tables in the Progress Dynamics Repository designed to store information about flows in an application. [Figure 19–2](#) shows the structure and relationships of the tables in this group.

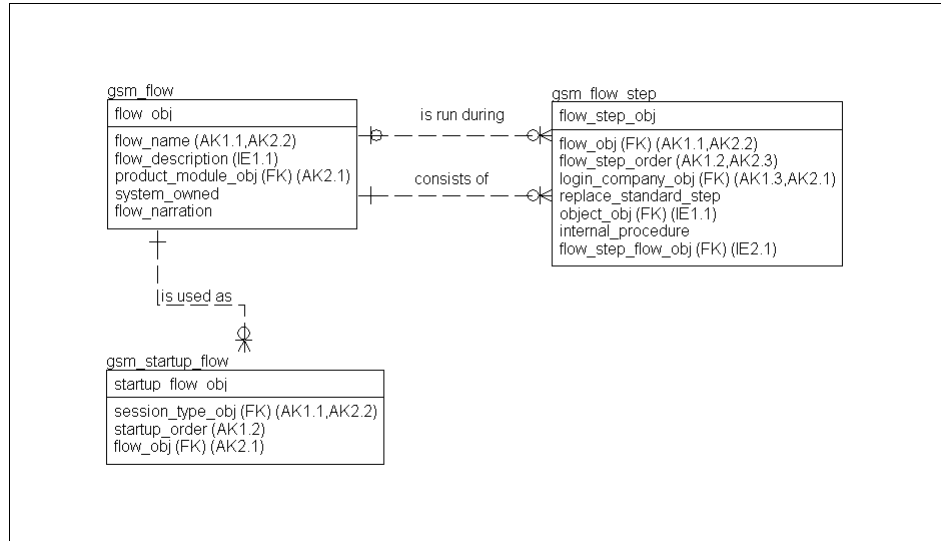


Figure 19–2: Flow group

The Flow group supports setting up sequences of actions that might need to change based on user variables. For example, a sales application that serves users in several regions might need to calculate sales tax differently based on region.

NOTE: This group of tables supports the planned event and flow functionality. These tables are liable to change. You should not use these tables in your applications.

The Flows group contains the following tables:

- gsm_flow table
- gsm_flow_step table
- gsm_startup_flow table

19.3 Profile group

This section covers the group of tables in the Progress Dynamics Repository designed to store information about profiles. [Figure 19–3](#) shows the structure and relationships of the tables in this group.

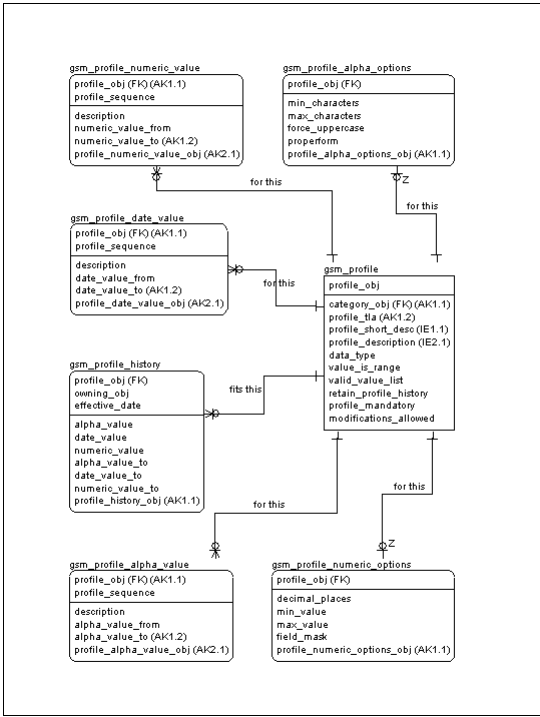


Figure 19–3: Profile group

NOTE: This group of tables supports the planned event and flow functionality. These tables are liable to change. You should not use these tables in your applications.

The Profiles group contains the following tables:

- gsm_profile table
- gsm_profile_alpha_options table
- gsm_profile_alpha_value table
- gsm_profile_date_value table
- gsm_profile_history table
- gsm_profile_numeric_options table
- gsm_profile_numeric_value table

19.4 Reporting Group

This section covers the group of tables in the Progress Dynamics Repository designed to store information about reporting.

Figure 19–4 shows the structure and relationships of the tables in this group.

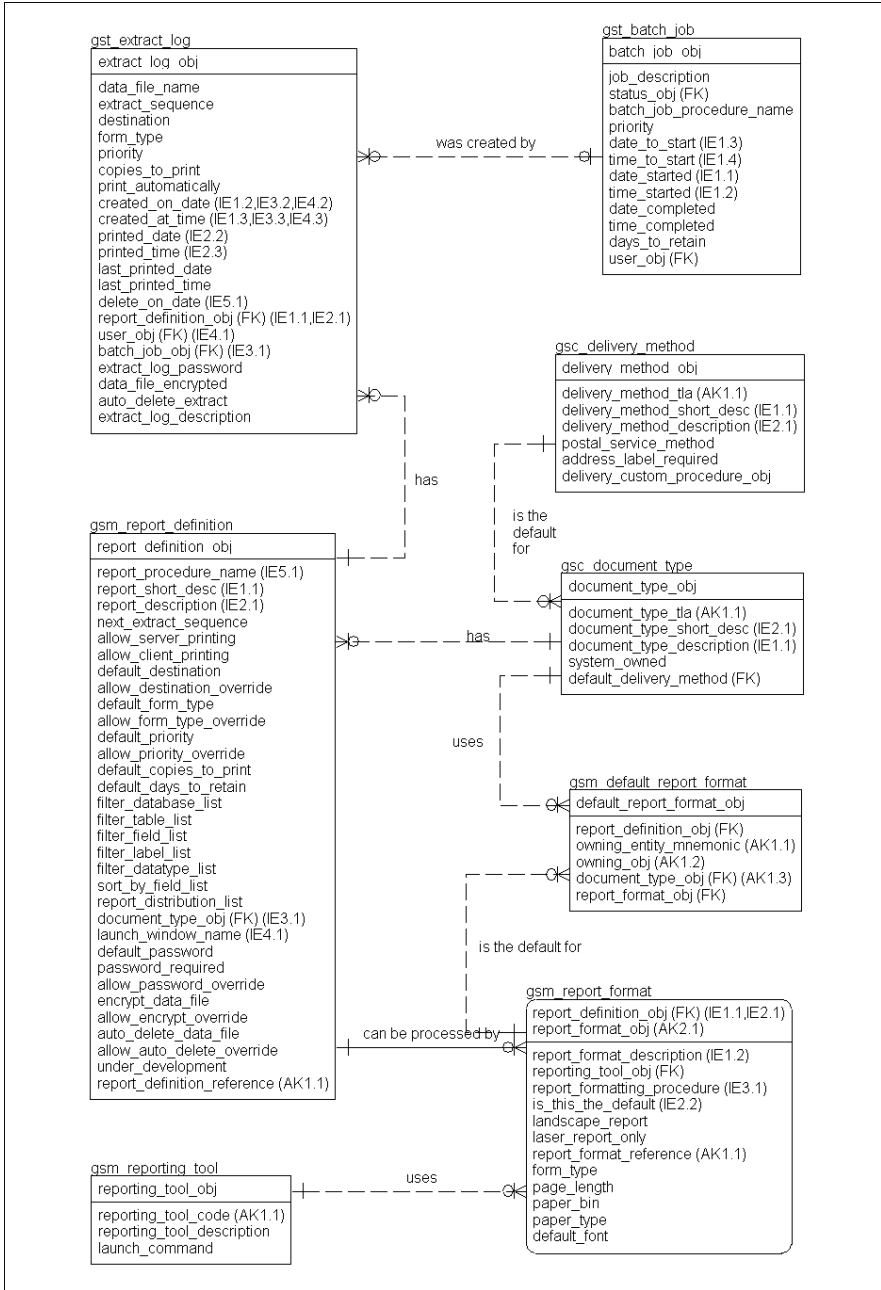


Figure 19–4: Reporting group

The Reporting group supports the addition of reporting solutions to the Progress Dynamics framework.

NOTE: This group of tables supports the planned event and flow functionality. These tables are liable to change. You should not use these tables in your applications.

The Reporting group contains the following tables:

- gsc_delivery_method table
- gsc_document_type table
- gsm_default_report_format table
- gsm_report_definition table
- gsm_report_format table
- gsm_reporting_tool table
- gst_batch_job table
- gst_extract_log table

Table Names and Acronyms

The Progress Dynamics framework makes extensive use of the acronyms (FLAs) as substitutes for the full names of Repository tables. The following tables are a quick reference for the tables and their FLAs:

- [FLA to table name reference](#)
- [Table name to FLA reference](#)

A.1 FLA to table name reference

Table A–1 lists the FLAs and their associated tables, alphabetically by FLA.

Table A–1: FLA to table name reference (1 of 6)

FLA	Table name
gsccp	gsc_custom_procedure
gscdc	gsc_default_code
gsydd	gsc_deploy_dataset
gsdde	gsc_dataset_entity
gsddm	gsc_delivery_method
gsddp	gsc_deploy_package
gsdds	gsc_default_set
gsddt	gsc_document_type
gsddu	gsc_default_set_usage
gsded	gsc_entity_display_field
gsdem	gsc_entity_mnemonic
gsdep	gsc_entity_mnemonic_procedure
gsder	gsc_error
gsdgc	gsc_global_control
gsdgd	gsc_global_default
gsdia	gsc_instance_attribute
gsdic	gsc_item_category
gsdlg	gsc_language
gscls	gsc_logical_service
gsclt	gsc_language_text

Table A–1: FLA to table name reference*(2 of 6)*

FLA	Table name
gscmm	gsc_multi_media_type
gscmt	gsc_manager_type
gscna	gsc_nationality
gscot	gsc_object_type
gscot	gsc_object_type
gscpc	gsc_profile_code
gscpf	gsc_profile_type
gscpm	gsc_product_module
gscpr	gsc_product
gscsc	gsc_security_control
gscsm	gsc_scm_tool
gscsn	gsc_next_sequence
gscsp	gsc_session_property
gscsq	gsc_sequence
gscst	gsc_service_type
gsctg	gsc_data_tag
gsmav	gsm_profile_alpha_value
gsmca	gsm_category
gsmcl	gsm_control_code
gsmcm	gsm_comment
gsmcr	gsm_currency
gsmcy	gsm_country

Table A–1: FLA to table name reference*(3 of 6)*

FLA	Table name
gsmdr	gsm_default_report_format
gsmeff	gsm_entity_field
gsmev	gsm_entity_field_value
gsmex	gsm_external_xref
gsmfd	gsm_filter_data
gsmff	gsm_field
gsmfi	gsm_filter_set
gsmfs	gsm_flow_step
gsmfw	gsm_flow
gsmga	gsm_group_allocation
gsmhe	gsm_help
gsmi	gsm_menu_structure_item
gsmig	gsm_login_company
gsmmi	gsm_menu_item
gsmmm	gsm_multi_media
gsmms	gsm_menu_structure
gsmnd	gsm_node
gsmnv	gsm_profile_numeric_value
gsmom	gsm_object_menu_structure
gsmpa	gsm_profile_alpha_options
gsmpd	gsm_profile_date_value
gsmpf	gsm_profile_data

Table A–1: FLA to table name reference*(4 of 6)*

FLA	Table name
gsmph	gsm_profile_history
gsmppn	gsm_profile_numeric_options
gsmpr	gsm_profile
gsmpp	gsm_physical_service
gsmra	gsm_range
gsmrd	gsm_report_definition
gsmre	gsm_reporting_tool
gsmrf	gsm_report_format
gsmrl	gsm_release
gsmrm	gsm_required_manager
gsmssc	gsm_server_context
gsmse	gsm_session_type
gsmssf	gsm_startup_flow
gsmsh	gsm_status_history
gsmss	gsm_security_structure
gsmst	gsm_status
gsmssv	gsm_session_service
gsmssx	gsm_scm_xref
gsmssy	gsm_session_type_property
gsmstd	gsm_tagged_data
gsmsti	gsm_translated_menu_item
gsmstl	gsm_translation

Table A–1: FLA to table name reference*(5 of 6)*

FLA	Table name
gsmtm	gsm_toolbar_menu_structure
gsmt0	gsm_token
gsmuc	gsm_user_category
gsmul	gsm_user_allocation
gsmus	gsm_user
gsmvp	gsm_valid_object_partition
gstad	gst_audit
gstbt	gst_batch_job
gstcs	gst_context_scope
gstdf	gst_dataset_file
gstdp	gst_deployment
gstel	gst_extract_log
gster	gst_error_log
gstph	gst_password_history
gstrl	gst_release_version
gstrv	gst_record_version
gstss	gst_session
rycap	ryc_attribute_group
rycat	ryc_attribute
rycav	ryc_attribute_value
ryccr	ryc_customization_result
ryccy	ryc_customization_type

Table A–1: FLA to table name reference*(6 of 6)*

FLA	Table name
rycla	ryc_layout
rycoi	ryc_object_instance
rycpa	ryc_page
rycpo	ryc_page_object
rycre	ryc_relationship
rycrf	ryc_relationship_field
rycri	ryc_ri_default
rycrt	ryc_render_type
rycsl	ryc_supported_link
rycsm	ryc_smartlink
rycso	ryc_smartobject
rycst	ryc_smartlink_type
rycue	ryc_ui_event
rymcz	rym_customization
rymdv	rym_data_version
rymwt	rym_wizard_tree
rytds	ryt_dbupdate_status

A.2 Table name to FLA reference

Table A–2 lists the Repository tables and their associated FLAs, alphabetically by table name.

Table A–2: Table name to FLA reference (1 of 6)

Table name	FLA
gsc_custom_procedure	gsccp
gsc_data_tag	gsctg
gsc_dataset_entity	gscode
gsc_default_code	gscdc
gsc_default_set	gs cds
gsc_default_set_usage	gs cdu
gsc_delivery_method	gs cdm
gsc_deploy_dataset	gs cdd
gsc_deploy_package	gs cdp
gsc_document_type	gs cdt
gsc_entity_display_field	gs ced
gsc_entity_mnemonic	gs cem
gsc_entity_mnemonic_procedure	gs cep
gsc_error	gs cer
gsc_global_control	gs cgc
gsc_global_default	gs cgd
gsc_instance_attribute	gs cia
gsc_item_category	gs cic
gsc_language	gs clg
gsc_language_text	gs clt

Table A–2: Table name to FLA reference*(2 of 6)*

Table name	FLA
gsc_logical_service	gscls
gsc_manager_type	gscmt
gsc_multi_media_type	gscmm
gsc_nationality	gscna
gsc_next_sequence	gscsn
gsc_object_type	gscot
gsc_object_type	gscot
gsc_product	gscpr
gsc_product_module	gscpm
gsc_profile_code	gscpc
gsc_profile_type	gscpf
gsc_scm_tool	gscsm
gsc_security_control	gscsc
gsc_sequence	gscsq
gsc_service_type	gscst
gsc_session_property	gscsp
gsm_category	gsmca
gsm_comment	gsmcm
gsm_control_code	gsmcl
gsm_country	gsmcy
gsm_currency	gsmcr
gsm_default_report_format	gsmdr

Table A–2: Table name to FLA reference*(3 of 6)*

Table name	FLA
gsm_entity_field	gsmef
gsm_entity_field_value	gsmev
gsm_external_xref	gsmex
gsm_field	gsmff
gsm_filter_data	gsmfd
gsm_filter_set	gsmfi
gsm_flow	gsmfw
gsm_flow_step	gsmfs
gsm_group_allocation	gsmga
gsm_help	gsmhe
gsm_login_company	gsmlg
gsm_menu_item	gsmmi
gsm_menu_structure	gsmms
gsm_menu_structure_item	gsmit
gsm_multi_media	gsmmm
gsm_node	gsmnd
gsm_object_menu_structure	gsmom
gsm_physical_service	gsmpr
gsm_profile	gsmpr
gsm_profile_alpha_options	gsmpr
gsm_profile_alpha_value	gsmav
gsm_profile_data	gsmpr

Table A–2: Table name to FLA reference*(4 of 6)*

Table name	FLA
gsm_profile_date_value	gsmpd
gsm_profile_history	gsmpd
gsm_profile_numeric_options	gsmpn
gsm_profile_numeric_value	gsmnv
gsm_range	gsmra
gsm_release	gsmrl
gsm_report_definition	gsmrd
gsm_report_format	gsmrf
gsm_reporting_tool	gsmre
gsm_required_manager	gsmrm
gsm_scm_xref	gsmxs
gsm_security_structure	gsmss
gsm_server_context	gsmsc
gsm_session_service	gsmss
gsm_session_type	gsmse
gsm_session_type_property	gsmssy
gsm_startup_flow	gsmssf
gsm_status	gsmst
gsm_status_history	gsmsh
gsm_tagged_data	gsmtd
gsm_token	gsmto
gsm_toolbar_menu_structure	gsmtm

Table A–2: Table name to FLA reference*(5 of 6)*

Table name	FLA
gsm_translated_menu_item	gsmti
gsm_translation	gsmtl
gsm_user	gsmus
gsm_user_allocation	gsmul
gsm_user_category	gsmuc
gsm_valid_object_partition	gsmvp
gst_audit	gstad
gst_batch_job	gstbt
gst_context_scope	gstcs
gst_dataset_file	gstdf
gst_deployment	gstdp
gst_error_log	gster
gst_extract_log	gstel
gst_password_history	gstph
gst_record_version	gstrv
gst_release_version	gstrl
gst_session	gstss
ryc_attribute	rycat
ryc_attribute_group	rycap
ryc_attribute_value	rycav
ryc_customization_result	ryccr
ryc_customization_type	ryccy

Table A–2: Table name to FLA reference*(6 of 6)*

Table name	FLA
ryc_layout	rycla
ryc_object_instance	rycoi
ryc_page	rycpa
ryc_page_object	rycpo
ryc_relationship	rycre
ryc_relationship_field	rycrf
ryc_render_type	rycrt
ryc_ri_default	rycri
ryc_smartlink	rycsm
ryc_smartlink_type	rycst
ryc_smartobject	rycso
ryc_supported_link	rycsl
ryc_ui_event	rycue
rym_customization	rymcz
rym_data_version	rymdv
rym_wizard_tree	rymwt
ryt_dbupdate_status	rytds

Index

C

Cardinality 1–2

Conventions

 .df files 1–4

 Repository 1–4

 Repository tables 1–5, 1–6

D

.df files

 latest applied –xvii, 1–4

 Repository naming conventions 1–4

E

ERwin 1–2

G

gsc_custom_procedure table 19–2

gsc_data_tag table 4–6

gsc_dataset_entity table 3–5

gsc_default_code table 4–7

gsc_default_set table 4–8

gsc_default_set_usage table 4–9

gsc_delivery_method table 19–7

gsc_deploy_dataset table 3–6

gsc_deploy_package table 3–8

gsc_document_type table 19–7

gsc_entity_display_field table 4–9

gsc_entity_mnemonic table 4–5

gsc_entity_mnemonic_procedure table
 19–2

gsc_error table 5–2

gsc_global_control table 6–5

gsc_global_default table 6–6

gsc_instance_attribute table 7–6

gsc_item_category table 7–8

gsc_language table 6–4

gsc_language_text table 6–7

gsc_logical_service table 13–5

gsc_manager_type table 13–6

gsc_multi_media_type table 9–3	gscgc table 6–5
gsc_nationality table 6–8	gscgd table 6–6
gsc_next_sequence table 12–4	gscia table 7–6
gsc_object_type table 10–12	gscic table 7–8
gsc_package_dataset table 3–8	gscig table 6–4
gsc_product table 8–3	gscls table 13–5
gsc_product_module table 8–4	gsclt table 6–7
gsc_profile_code table 17–3	gscmm table 9–3
gsc_profile_type table 17–4	gscmt table 13–6
gsc_scm_tool table 3–9	gscna table 6–8
gsc_security_control table 11–4	gscot table 10–12
gsc_sequence table 12–2	gscpc table 17–3
gsc_service_type table 13–8	gscpd table 3–8
gsc_session_property table 13–9	gscpf table 17–4
gsccp table 19–2	gscpm table 8–4
gscdc table 4–7	gscpr table 8–3
gsydd table 3–6	gscsc table 11–4
gscode table 3–5	gscsm table 3–9
gscdm table 19–7	gscsn table 12–4
gsmdp table 3–8	gscsp table 13–9
gsdds table 4–8	gscsq table 12–2
gsddt table 19–7	gscst table 13–8
gsddu table 4–9	gsctg table 4–6
gsced table 4–9	gsm_category table 14–3
gschem table 4–5	gsm_comment table 9–4
gscep table 19–2	gsm_control_code table 4–11
gsцер table 5–2	gsm_country table 6–9

gsm_currency table 6–10	gsm_profile_numeric_value table 19–5
gsm_default_report_format table 19–7	gsm_range table 11–7
gsm_entity_field table 4–12	gsm_release table 3–10
gsm_entity_field_value table 4–12	gsm_report_definition table 19–7
gsm_external_xref table 4–13	gsm_report_format table 19–7
gsm_field table 11–5	gsm_reporting_tool table 19–7
gsm_filter_data table 4–15	gsm_required_manager table 13–11
gsm_filter_set table 4–16	gsm_scm_xref table 3–11
gsm_flow table 19–3	gsm_security_structure table 11–8
gsm_flow_step table 19–3	gsm_server_context table 13–12
gsm_group_allocation table 11–6	gsm_session_service table 13–13
gsm_help table 18–4	gsm_session_type table 13–4
gsm_login_company table 18–5	gsm_session_type_property table 13–14
gsm_menu_item table 7–4	gsm_startup_flow table 19–3
gsm_menu_structure table 7–8	gsm_status table 14–4
gsm_menu_structure_item table 7–10	gsm_status_history table 14–6
gsm_multi_media table 9–5	gsm_tagged_data table 4–17
gsm_node table 15–3	gsm_token table 11–9
gsm_object_menu_structure table 7–10	gsm_toolbar_menu_structure table 7–12
gsm_physical_service table 13–10	gsm_translated_menu_item table 7–12
gsm_profile table 19–5	gsm_translation table 6–11
gsm_profile_alpha_options table 19–5	gsm_user table 16–4
gsm_profile_alpha_value table 19–5	gsm_user_allocation table 16–6
gsm_profile_data table 17–4	gsm_user_category table 16–8
gsm_profile_date_value table 19–5	gsm_valid_object_partition table 13–15
gsm_profile_history table 19–5	gsmav table 19–5
gsm_profile_numeric_options table 19–5	gsmca table 14–3

gsmcl table 4–11	gsmpn table 19–5
gsmcm table 9–4	gsmprr table 19–5
gsmcr table 6–10	gsmpy table 13–10
gsmcy table 6–9	gsmra table 11–7
gsmdr table 19–7	gsmdr table 19–7
gsmef table 4–12	gsmre table 19–7
gsmev table 4–12	gsmrfr table 19–7
gsmex table 4–13	gsmlr table 3–10
gsmfd table 4–15	gsmlr table 13–11
gsmff table 11–5	gsmsc table 13–12
gsmfi table 4–16	gsmsc table 13–4
gsmfs table 19–3	gsmsf table 19–3
gsmfw table 19–3	gsmsr table 14–6
gsmga table 11–6	gsmsr table 11–8
gsmhe table 18–4	gsmsr table 14–4
gsmit table 7–10	gsmsv table 13–13
gsmlg table 18–5	gsmsx table 3–11
gsmmi table 7–4	gsmsy table 13–14
gsmmm table 9–5	gsmtd table 4–17
gsmps table 7–8	gsmti table 7–12
gsrnd table 15–3	gsmtl table 6–11
gsrnv table 19–5	gsmtm table 7–12
gsmom table 7–10	gsmtt table 11–9
gsmpa table 19–5	gsmtt table 16–8
gsmpd table 19–5	gsmtt table 16–6
gsmpf table 17–4	gsmtt table 16–4
gsmpg table 19–5	gsmtt table 13–15

gst_audit table 18–2
gst_batch_job table 19–7
gst_context_scope table 13–15
gst_dataset_file table 3–13
gst_deployment table 3–4
gst_error_log table 5–4
gst_extract_log table 19–7
gst_password_history table 16–9
gst_record_version table 3–14
gst_release_version table 3–15
gst_session table 13–17
gstad table 18–2
gstbt table 19–7
gstcs table 13–15
gstdf table 3–13
gstdp table 3–4
gstel table 19–7
gster table 5–4
gstph table 16–9
gstrl table 3–15
gstrv table 3–14
gstss table 13–17

I

IE notation 1–2
 cardinality 1–2

M

Model 1–2
 ERwin 1–2
 IE notation 1–2

O

_obj fields
 Repository conventions 1–6

R

Repository
 .df file conventions 1–4
 _obj fields 1–6
 conventions 1–4
 FLAs 1–6
 model 1–2
 table conventions 1–5, 1–6
ryc_attribute table 10–15
ryc_attribute_group table 10–16
ryc_attribute_value table 10–17
ryc_customization_result table 2–3
ryc_customization_type table 2–4
ryc_layout table 10–19
ryc_object_instance table 10–20
ryc_page table 10–22
ryc_relationship table 4–18
ryc_relationship_field table 4–20
ryc_render_type table 2–5
ryc_ri_default table 4–21
ryc_smartlink table 10–23
ryc_smartlink_type table 10–24

ryc_smartobject table 10–10

ryc_supported_link table 10–25

ryc_ui_event table 10–26

rycap table 10–16

rycat table 10–15

rycav table 10–17

ryccr table 2–3

ryccy table 2–4

rycla table 10–19

rycoi table 10–20

rycpa table 10–22

rycre table 4–18

rycrf table 4–20

rycri table 4–21

rycrt table 2–5

rycsl table 10–25

rycsm table 10–23

rycso table 10–10

rycst table 10–24

rycue table 10–26

rym_customization table 2–6

rym_data_version table 10–28

rym_wizard_tree table 15–4

rymcz table 2–6

rymdv table 10–28

rymwt table 15–4

ryt_dbupdate_status table 3–16

rytds table 3–16

S

Schema definition files

Repository naming conventions 1–4

T

Tables

FLAs 1–6

Repository _obj fields 1–6

Repository conventions 1–5, 1–6