MOST

Media **O**riented **S**ystems **T**ransport
Multimedia and Control
Networking Technology

MOST Dynamic Specification
Rev 3.0.2
10/2012

Legal Notice

COPYRIGHT

LICENSE DISCLAIMER

Nothing on any MOST Cooperation Web Site, or in any MOST Cooperation document, shall be construed as conferring any license under any of the MOST Cooperation or its members or any third party's intellectual property rights, whether by estoppel, implication, or otherwise.

CONTENT AND LIABILITY DISCLAIMER

MOST Cooperation or its members shall not be responsible for any errors or omissions contained at any MOST Cooperation Web Site, or in any MOST Cooperation document, and reserves the right to make changes without notice. Accordingly, all MOST Cooperation and third party information is provided "AS IS". In addition, MOST Cooperation or its members are not responsible for the content of any other Web Site linked to any MOST Cooperation Web Site. Links are provided as Internet navigation tools only.

MOST COOPERATION AND ITS MEMBERS DISCLAIM ALL WARRANTIES WITH REGARD TO THE INFORMATION (INCLUDING ANY SOFTWARE) PROVIDED, INCLUDING THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT. Some jurisdictions do not allow the exclusion of implied warranties, so the above exclusion may not apply to you.

In no event shall MOST Cooperation or its members be liable for any damages whatsoever, and in particular MOST Cooperation or its members shall not be liable for special, indirect, consequential, or incidental damages, or damages for lost profits, loss of revenue, or loss of use, arising out of or related to any MOST Cooperation Web Site, any MOST Cooperation document, or the information contained in it, whether such damages arise in contract, negligence, tort, under statute, in equity, at law or otherwise.

FEEDBACK INFORMATION

Any information provided to MOST Cooperation in connection with any MOST Cooperation Web Site, or any MOST Cooperation document, shall be provided by the submitter and received by MOST Cooperation on a non-confidential basis. MOST Cooperation shall be free to use such information on an unrestricted basis.

TRADEMARKS

MOST Cooperation and its members prohibit the unauthorized use of any of their trademarks. MOST Cooperation specifically prohibits the use of the MOST Cooperation LOGO unless the use is approved by the Steering Committee of MOST Cooperation.

SUPPORT AND FURTHER INFORMATION

For more information on the MOST technology, please contact:

MOST Cooperation
Administration
Bannwaldallee 48
D-76185 Karlsruhe
Germany

Tel: (+49) (0) 721 966 50 00

E-mail: contact@mostcooperation.com
Web: www.mostcooperation.com

Contents

Document History

Changes MOST Dynamic Specification Rev 3.0.1 to MOST Dynamic Specification Rev 3.0.2

Change Ref.	Section	Changes
3V02_001	All	– Correction of minor clerical errors.
3V02_002	Bibliography	– Updated document version numbers.
3V02_003	3.1.3	– Changed the precondition in the table to “NetInterface Init” – Using “Initialize Central Registry” instead of “delete”. – Using single NotOK message instead of reference to NM_Gen_SystemConfigurationUpdate. – Removed timer $t^{WaitBeforeScan}$ because it is not started. – Split Decentral Registry action into two actions. “Derive NodeAddress” remains inside the OPT box, “Initialize Decentral Registry” after the OPT inline expression.
3V02_004	3.1.5	– Changed the precondition in the table to “Normal Operation” – In the table under “Remarks”, “This process is not affected by an NCE.” becomes “In this scenario, an NCE does not occur.” because an NCE would terminate this scenario. – In the table, under “Description”, deleted “If a node registration is correct, it will always be entered into the Central Registry independent of the state of the network.” because this depends on the behavior of the Slave. – To “when (Error in registration)—Duplicate or invalid NodeAddress.” added “or duplicate or invalid NetBlock.InstID”. – “Delete Decentral Registry and derive NodeAddress” - divide into two actions because different parts perform those actions. – “when (any InstID is invalid or duplicate)” becomes “when (any application InstID is invalid or duplicate)” – “Go to SystemState OK.”—the following setting condition was redundant. Removed setting condition and changed comment to “Successful configuration update leads to SystemState OK.”
3V02_005	3.1.6	– Changed “Scan generated a fatal error.” to “Scan or single request generated an error.”
3V02_006	3.1.7	– In the table, under “Description” changed “When an NCE is detected, the whole network is rescanned.” to “When an NCE is detected, the NetworkMaster interrupts its action and scans the network.”
3V02_007	3.2.1	– In the table under “Remarks”, added “Spontaneous registrations do not occur.”
3V02_008	3.2.3.2	– In the table under “Remarks”, added “The concrete values for FBlockID=0x22 and InstID=0x01/0x02 are examples.”
3V02_009	3.2.3.3	– Changed “when (NWM_supports_immediate_notification = true)” to “false” in the second case. – Using a PAR inline expression so that it looks more like the other MSCs and no unnecessary order is imposed. – In the table under “Remarks”, added “The concrete values for FBlockID=0x22 and InstID=0x01/0x02 are examples.”
3V02_010	3.2.3.4	– Changed “NetworkSlave_2 has recalculated its NodePosition.” to “Set logical node address.” and made the text element an action.
3V02_011	3.2.3.5	– In the table, under “Description”, changed “The NetworkMaster will allow this node and continue to request its configuration.” to “The NetworkMaster continues to request its configuration.” – Removed “Allow missing node” action – Deleted comment “Wait for tDelayCfgRequest to expire.” – Added alternative with “NetworkSlave_2 registers no new FBlocks.” similar to “NetworkSlave_2 registers new FBlocks correctly.” The NetworkMaster reacts by sending NewExt with an empty list.
3V02_012	3.2.3.6	– Changed “message to lost” so that it contains “0x102, FBlockID = 0x22 InstID = 0x01” as registered. – Removed action “Allow missing node”

Change Ref.	Section	Changes
3V02_013	3.2.3.7	– Deleted both occurrences of “Allow node, treat as notresponding”
3V02_014	3.2.3.8	– Deleted both occurrences of “Allow node, treat as notresponding”
3V02_015	3.2.3.9	– Renamed the section to “Node causes NotOK too many times” – Changed the name of the scenario to NM_Sc_Scan_Node_Causes_NotOK_Too_Many_Times – In the description, replaced “NetworkSlave_2 causes System State NotOK three times in succession.” with “NetworkSlave_2 causes System State NotOK the maximum allowed number of times.”
3V02_016	3.2.3.11	– Changed the precondition in the table to “NetInterface Normal Operation”
3V02_017	3.2.3.12	– Changed the precondition in the table to “NetInterface Normal Operation” – Changed "Thus, no message is sent after the System Scan is complete." to "Thus, no Configuration.Status message is sent after the System Scan is complete."
3V02_018	3.4.1	– Removed guarding condition with reference to SystemCommunicationInit
3V02_019	3.4.2	– Changed the precondition in the table to “NetInterface Init” – "SystemState is OK when Configuration.Status(NewExt) is received." becomes action "Set System State OK". – "SystemState is OK when Configuration.Status(Invalid) is received." becomes action "Set System State OK". – Removed reference to SystemCommunicationInit – Replaced reference to NS_Gen_Reset by action "Perform Network Configuration Reset" and add a comment "see MOST Specification for individual actions"
3V02_020	3.4.3	– Change the precondition in the table to “NetInterface Normal Operation” – Remove reference to SystemCommunicationInit. – Remove Configuration.Status(OK) because this cannot be sent in System State OK. – Add a remark for Configuration.Status(Invalid) that own configuration invalid handling is not considered here. – Extract messages that are independent of the system state and put them outside the guarding conditions. – For FBlockIDs.SetGet, Error is an alternative to Status. Add Error message.
3V02_021	-	– From Dynamic Specification Rev. 3.0.2, removed section 3.4.4 Reset because it contained no communication and can be replaced by an action in the referencing MSCs.
3V02_022	3.4.4	– Using "0x42" for FBlockID and "_" for InstID; added comment: "Request TVTuner" – In the condition “when (CR_Contains_NWSlave_2 = false)”, now using "NWSlave_1" instead. – In “CentralRegistry.Error(ErrorCode='0x07', ErrorInfo='0x01, 0x0n.01')” changed ErrorInfo="0x01, _". – “CentralRegistry.Status(FBlockInfoList='RxTxLog=NetworkSlave_1,FBlockID=0x0n,InstID=0x01')” - changed to "0x42, InstID=0x01"; added comment "found FBlockID 0x42, InstID 0x01".
3V02_023	3.5.1.1	– Set the precondition in the table to “NetInterface Init” – Remove reference to SystemCommunicationInit – Add OPT inline expression, containing action "Build Decentral Registry"
3V02_024	3.5.1.2	– Set the precondition in the table to “NetInterface Init” – Replaced reference to NS_Gen_Reset by action "Perform Network Configuration Reset" - comment "see MOST Specification for individual actions" – Removed reference to SystemCommunicationInit
3V02_025	3.5.2.1	– In “CentralRegistry.Get(FBlockID='0x31', InstID='0x01')”, now using "InstID='_'"
3V02_026	4.2.1.1	– Using ErrorAck with SenderHandle instead of Error.
3V02_027	4.2.1.2	– Using ErrorAck with SenderHandle instead of Error.
3V02_028	4.2.1.3	– Using ErrorAck with SenderHandle instead of Error.

Change Ref.	Section	Changes
3V02_029	4.2.1.4	– Using ErrorAck with SenderHandle instead of Error.
3V02_030	4.2.1.5	– Using ErrorAck with SenderHandle instead of Error.
3V02_031	4.2.1.5.1	– Using ErrorAck with SenderHandle instead of Error.
3V02_032	4.2.1.5.2	– Using ErrorAck with SenderHandle instead of Error.
3V02_033	-	– Replaced section 5.2, including the MSC, with “Switching on the network is described in the MOST Specification.” because the MSC did not contain communication.
3V02_034	5.2	– Added “Prior Condition” in table: “NetInterface Normal Operation” – Added setting condition SystemState_NotOK after message ShutDown.Start(Control='Execute').
3V02_035	5.3	– Added “Prior Condition”: “NetInterface Normal Operation”
3V02_036	5.4	– Added “Prior Condition” in table: “NetInterface Normal Operation” – SystemCommunicationInit; becomes “perform SystemCommunicationInit”.
3V02_037	5.5	– Added “Prior Condition”: “NetInterface Normal Operation” – Changed comment from “Temperature near critical limit” to “Temperature near device-specific critical limit.” – Made EXC inline expression parallel to everything else so that critical temperature shuts down anytime.
3V02_038	5.6	– Made EXC inline expression parallel to everything else so that critical temperature shuts down anytime.
3V02_039	-	– Removed chapter “Function Blocks”.

Changes MOST Dynamic Specification Rev 3.0 to MOST Dynamic Specification Rev 3.0.1

Change Ref.	Section	Changes
3V01_001	All	– Correction of typos and minor clerical errors. – Removed “Communication Partner” row from all MSC tables because this information is self-evident. – MSC comments are now contained in “comment brackets” rather than referenced and listed below the MSCs.
3V01_002	Bibliography	– Added GeneralFBlock to document list.
3V01_003	3	– Removed misleading note that stated that the information in the Central Registry was not distributed.
3V01_004	–	– Removed section “System States” because this is already contained in the MOST Specification.
3V01_005	3.1.1	– Changed description of variable doRequest. – Added more variables to Table 3-1.
3V01_006	3.1.3	– Removed comment regarding addresses being static, stored, or calculated. Addresses are no longer stored and the other two are obvious. – $t_{WaitBeforeScan}$, which previously was a timer, is now a timing condition.
3V01_007	3.1.5	– Added $t_{DelayCfgRequest}$ timeout before “doRequest” is set to true.
3V01_008	3.2.1	– Removed comment regarding addresses being static, stored, or calculated. Addresses are no longer stored and the other two are obvious. – $t_{WaitBeforeRescan}$, which previously was a timer, is now a timing condition.
3V01_009	3.2.2.1	– Removed remark regarding addresses being stored, which no longer is the case. – Added MSC reference to NM_Sc_Scan_SystemStatus_To_OK. – $t_{WaitBeforeScan}$, which previously was a timer, is now a timing condition.
3V01_010	–	– Removed section “Initial Scan System State NotOK to OK” because of redundancy.
3V01_011	3.2.3.1	– Renamed section from “Normal Scan without implications” to “Normal Scan”. – Renamed MSC to NM_Sc_Scan_SystemState_To_OK.

Change Ref.	Section	Changes
3V01_012	3.2.3.2	– Renamed MSC to NM_Sc_Scan_InstID_Mismatch_SystemState_OK.
3V01_013	3.2.3.3	– Renamed MSC to NM_Sc_Scan_InstID_Collision_SystemState_To_OK.
3V01_014	3.2.3.6	– Included “own configuration invalid” handling in MSC.
3V01_015	3.2.3.13	– Using Configuration.Status(NewExt) instead of New.
3V01_016	3.3	– New section “Variables used in NetworkSlave MSCs”.
3V01_017	3.4.2	– Removed comment regarding addresses being static, stored, or calculated. Addresses are no longer stored and the other two are obvious.
3V01_018	3.4.3	– Removed comment regarding addresses being static, stored, or calculated. Addresses are no longer stored and the other two are obvious.
3V01_019	3.4.4	– Removed comment regarding addresses being static, stored, or calculated. Addresses are no longer stored and the other two are obvious.
3V01_020	3.4.5	– Added condition for sending CentralRegistry.Error to a NetworkSlave.
3V01_021	3.5.1.1	– Removed comment regarding addresses being static, stored, or calculated. Addresses are no longer stored and the other two are obvious.
3V01_022	3.5.1.2	– Removed comment regarding addresses being static, stored, or calculated. Addresses are no longer stored and the other two are obvious.
3V01_023	–	– Removed section “Logical Model of Connection Management” because the information contained here was not used.
3V01_024	4.1	– Added more variables to Table 4-1.
3V01_025	4.2	– Rephrased so that the notion of a “familiar system” is no longer used.
3V01_026	4.2.1.1	– Renamed MSC to CM_Gen_Connect_StartResultAck.
3V01_027	4.2.1.2	– Renamed MSC to CM_Gen_SourceActivity_StartResultAck.
3V01_028	4.2.1.4	– Added remark regarding SourceActivity function. – Renamed MSC to CM_Gen_Allocate_StartResultAck.
3V01_029	4.2.1.5.1	– Renamed MSC to CM_Gen_DisConnect_StartResultAck.
3V01_030	4.2.1.5.2	– Renamed MSC to CM_Gen_DeAllocate_StartResultAck. – Added remark that the source stops routing and removes its channels.
3V01_031	4.3	– Added section “DiscreteFrame Isochronous Connection Handling”.
3V01_032	–	– Removed section “General Sink MSCs” because it was mostly redundant. Relevent part were moved to the Connection Management section.
3V01_033	–	– Removed section “General Source MSCs” because it was mostly redundant. Relevent part were moved to the Connection Management section.
3V01_034	5.2	– Removed use of PermissionToWake function, which no longer exists.
3V01_035	5.3	– Extended MSC to consider the use cases “cancel shutdown” and “force shutdown”. – Replaced $t_{Suspend}$ with $t_{WaitSuspend}$.
3V01_036	5.4	– Replaced $t_{Suspend}$ with $t_{WaitSuspend}$. – ShutDown function uses OPTypes without SenderHandle again.
3V01_037	5.5	– ShutDown function uses OPTypes without SenderHandle again. – Added SystemCommunicationInit action after receiving wake-up from device shutdown.
3V01_038	5.6	– ShutDown function uses OPTypes without SenderHandle again. – theta_Critical replaces theta_Dead – PermissionToWake is not used anymore.
3V01_039	5.7	– ShutDown function uses OPTypes without SenderHandle again. – PermissionToWake is not used anymore.
3V01_040	8	– Removed GSI and GSO from Table 8-1 because these are no longer used.

Changes MOST Dynamic Specification Rev 1.3 to MOST Dynamic Specification Rev 3.0

Change Ref.	Section	Changes
3V0_001	General	– Replaced non-Ack OPTypes with Ack-OPTypes because MOST Specification 3.0 recommends the use of Ack OPTypes. – Changed instance name from AudioDiskPlayer to AudioDiscPlayer for consistency. – Renamed function BuildSyncConnection to BuildConnection to match FBlock Bundle 3.0. – Fixed clerical errors
3V0_002	3.1	– Updated state chart to match MOST Specification Rev. 3.0.
3V0_003	3.2.6 3.3.2	– Configuration.Status Broadcast was sent to the environment. However, the MSC standard requires that the resulting gate message is connected to every reference to this MSC. To prevent this, the broadcast is no longer forwarded to the environment.
3V0_004	3.3.3.1	– "when (Init Ready detected)" is an event in at least one other MSC. Using to event here now, as well. – Removed mention of Decentral Registry for initial scan because the concept of a stored Decentral Registry was abandoned.
3V0_005	3.3.3.2	– Removed mention of Decentral Registry for initial scan because the concept of a stored Decentral Registry was abandoned.
3V0_006	-	– Removed obsolete section “Initial Scan with Node Not Responding”.
3V0_007	-	– Removed obsolete section “Error during Initial Scan with a stored Central Registry”
3V0_008	-	– Secondary nodes are no longer supported. Deleted this MSC.
3V0_009	3.3.4.2	– Configuration.Status(New) no longer used. Replaced with NewExt.
3V0_010	3.3.4.3	– Configuration.Status(New) no longer used. Replaced with NewExt.
3V0_011	3.3.4.5	– Configuration.Status(New) no longer used. Replaced with NewExt.
3V0_012	3.3.4.6	– Configuration.Status(New) no longer used. Replaced with NewExt. – Updated DeltaFBlocklist to match FBlock Bundle 3.0.
3V0_013	3.3.4.7	NM_Sc_Scan_Node_Reporting_Error_In_NotOK: – Configuration.Status(New) no longer used. Replaced with NewExt. – Secondary nodes are no longer supported; therefore, renamed this MSC.
3V0_014	3.3.4.8	NM_Sc_Scan_Node_Reporting_Error_In_OK: – Configuration.Status(New) no longer used. Replaced with NewExt. – Secondary nodes are no longer supported; therefore, renamed this MSC.
3V0_015	3.3.4.12	– Configuration.Status(New) no longer used. Replaced with NewExt.
3V0_016	3.3.4.14	– Configuration.Status(New) no longer used. Replaced with NewExt. – Configuration.Status Broadcast is sent to the environment. However, the MSC standard requires that the resulting gate message is connected to every reference to this MSC. To prevent this, do not forward to environment.
3V0_017	3.4.1	– Contains reference to secondary nodes. Secondary nodes are no longer supported. Deleted corresponding path in the MSC.
3V0_018	3.4.2	– Configuration.Status(New) no longer used. Replaced with NewExt. – Contained Secondary Node paths. Removed them.
3V0_019	3.4.3	NS_Gen_RunningNode: – Configuration.Status(New) no longer used. Replaced with NewExt. – Changed guarding condition that checkes whether the node is a Primary node. – Primary nodes are now called MOST nodes. Renamed MSC
3V0_020	-	– NS_Gen_RunningSecondary: Secondary nodes are no longer supported. Deleted this MSC.
3V0_021	3.4.4	– Removed requirement for the source to route zeros.
3V0_022	3.4.5	– NS_Gen_Communicate: "USERDEF.Msg" between two MOST nodes. Changed to an exemplary message contained in FBlock Bundle 3.0.
3V0_023	-	– Deleted obsolete section “NetworkSlave changes NodeAddress in SystemState OK”
3V0_024	-	– Removed “Introduction” section of Connection Management because it described the difference between SourceConnect and Allocate; however, SourceConnect is no longer available.

Change Ref.	Section	Changes
3V0_025	4.2	– Removed SourceType variable because no distinction is required between SourceConnect and Allocate anymore. SourceConnect is no longer available.
3V0_026	4.3	– SourceConnect no longer available. Removed corresponding descriptions.
3V0_027	-	– Removed CM_Gen_SC_BuildSyncConnection because SourceConnect is no longer supported.
3V0_028	-	– CM_Gen_SC_SourceInfo_Get removed because this MSC was only referenced by SourceConnect scenarios, which are no longer supported.
3V0_029	-	– CM_Gen_SC_SourceConnect_StartResult: SourceConnect approach no longer supported. Deleted this MSC.
3V0_030	4.3.1.1	– CM_Gen_Connect_StartResult: Updated parameter list for Connect to match FBlock Bundle 3.0.
3V0_031	4.3.1.3	– CM_Gen_BuildConnection: Renamed MSC and removed SourceConnect paths. SourceConnect is no longer supported.
3V0_032	-	– CM_Gen_M_Get_SourceType: Requesting SourceType refers to SourceConnect. Deleted entire MSC.
3V0_033	-	– CM_Gen_SC_SourceInfo_Get: MSC deleted because SourceConnect is no longer supported.
3V0_034	4.3.1.4	CM_Gen_Allocate_StartResult – Allocate contained old parameter list. Updated. – Renamed MSC because “mixed systems” no longer exist after SourceConnect became obsolete.
3V0_035	4.3.1.5	– Changed description so that SourceConnect is no longer mentioned.
3V0_036	-	– CM_Gen_SC_SourceDisConnect_StartResult: SourceConnect approach no longer supported. Deleted this MSC.
3V0_037	4.3.1.5.2	CM_Gen_DeAllocate_StartResult: – SourceConnect approach no longer supported. Removed the corresponding path. – Renamed MSC.
3V0_038	-	– CM_Sc_SC_Known: SourceConnect no longer supported. Deleted MSC.
3V0_039	-	– CM_Sc_SC_Unknown: SourceConnect approach no longer supported. Deleted entire MSC.
3V0_040	-	– CM_Sc_Mixed_System: SourceConnect approach no longer supported. Deleted entire MSC.
3V0_041	-	– CM_Sc_RemoveConnection: SourceConnect approach no longer supported.Deleted entire MSC.
3V0_042	-	– CM_Gen_SC_CleanUp: SourceConnect approach is no longer supported. Deleted this MSC.
3V0_043	-	– CM_Sc_SourceDrop: Deleted MSC. Already not supported for MOST 50.
3V0_044	4.4.1	– Removed description of use of Ack OPTypes. Those are no longer recommended.
3V0_045	4.4.1.1	– CM_Gen_CleanUp: Removed SourceConnect path. Renamed MSC. Fixed dangling reference.
3V0_046	4.4.2.1	– CM_Sc_SinkDrop: SourceConnect approach no longer supported. Removed corresponding path.
3V0_047	-	– GSO_Ge_SourceConnect: SourceConnect approach no longer supported. Deleted this MSC.
3V0_048	-	– GSO_Ge_SourceDisconnect: SourceConnect approach no longer supported. Deleted this MSC.
3V0_049	-	– Removed empty section “Extended ConnectionMaster General MSCs”.
3V0_050	-	– Removed empty section “Extended ConnectionMaster Scenarios”.
3V0_051	4.4.2.2	– New MSC CM_Sc_Source_Malfunction.
3V0_052	5.3	– PM_Gen_Network_Shutdown: In the referencing MSC PM_Gen_Overtemp_Shutdown there are three NetBlock instances. Using one instance, matching it with an instance parameter to the referenced MSC.
3V0_053	-	– Removed chapter “Generic Management of Audio (Synchronous data)”.

Change Ref.	Section	Changes
3V0_054	6.2.5	– ADP_Sc_Search: "SearchDirection" not defined for Enum in DeckStatus. It is either forward or backward. Using forward.
3V0_055	6.2.6	– ADP_Sc_NewTrackUnchanged: User_selects_same_track is missing the USERDEF alias name. Added USERDEF.
3V0_056	6.2.7	– ADP_Sc_StartScanDisc: User_pushes_the_scan_button is missing the USERDEF alias name. Added USERDEF.
3V0_057	6.2.8	– ADP_Sc_StopScanDisc: MOST.Scan_disc_ends is not a MOST message. Changed to USERDEF.
3V0_058	6.2.9	– ADP_Sc_StartRandom: User_selects_random is missing the USERDEF alias name. Added USERDEF.
3V0_059	6.2.10	– ADP_Sc_StopRandom: User_stops_random is missing the USERDEF alias name. Added USERDEF.
3V0_060	6.2.11	– ADP_Sc_StartRepeatTrack: User_selects_repeat_track is missing the USERDEF alias name. Added USERDEF.
3V0_061	6.2.13	– ADP_Sc_StopRepeat: User_turns_repeat_off is missing the USERDEF alias name. Added USERDEF.
3V0_062	-	– Removed section “Connection Management Naming Conventions” because the content depended on the use of the SourceConnect approach.

Changes MOST Dynamic Specification Rev 1V2 to MOST Dynamic Specification Rev 1V3

Change Ref.	Section	Changes
1V3_001	General	– Minor spelling and grammar corrections. – Fixed parameters to match Function Library. – Modified timer names to match the MOST Specification. – Replaced "synchronous" with "streaming" to match the terminology of the MOST Specification. – Where applicable, added note that Secondary Nodes are not supported by MOST50. – Removed elements/notes that mentioned the stored Registry. The concept of a stored Central/Decentral Registry is no longer supported by the MOST Specification. – Replaced NetOn event with Init Ready event. – Replaced MOST.NCE with USERDEF.NCE because the NCE is not contained in any function catalog. – Changed channel info from "AudioAmplifier" to "MOST" to match other MSCs that belong to the Dynamic Specification. – Replaced "all bypass" with "bypass".
1V3_002	3.1	– Updated System States diagram to include Shutdown.Start(Execute) transition.
1V3_003	3.2.2	NM_Gen_Startup – Added end node to HMSC. – Removed NetOn setting condition because the NetOn state is only reached after Init Ready is received.
1V3_004	3.2.3	NM_Gen_Init – Replaced *when NetOn* condition with reception of Init Ready event.
1V3_005	3.2.7	NM_Gen_ProcessNCE – Changed single cast NCE into a broadcast message.
1V3_006	3.3.3.1	NM_Sc_Initial_Scan_CR_Not_Stored_SystemState_NotOK – Modified and renamed to NM_Sc_Initial_Scan_NoNodeAddress_SystemState_NotOK because Central Registry is no longer stored. – Renamed section to “Initial Scan without Node Address”.
1V3_007	3.3.3.2	NM_Sc_Initial_Scan_CR_Stored_SystemState_NotOK_To_OK – Modified and renamed to NM_Sc_Initial_Scan_SystemState_NotOK_To_OK because Central Registry is no longer stored. – Renamed section to “Initial Scan System State NotOK to OK”
1V3_008	-	NM_Sc_Initial_Scan_CR_Stored_Node_Not_Responding – MSC removed because the MOST Specification does not support a stored Central Registry anymore.

Change Ref.	Section	Changes
1V3_009	3.3.3.4	NM_Sc_Initial_Scan_CR_Stored_Registration_Error – MSC removed because the MOST Specification does not support a stored Central Registry anymore.
1V3_010	3.4.1	NS_Gen_Startup – Added HMSC End element. – Added note that secondary nodes are not supported by MOST50. – Removed NetOn condition because Init Ready event was received yet.
1V3_011	3.4.2	NS_Gen_Init – Renamed NetOn_event to InitReady event. NetOn event is no longer used.
1V3_012	-	NM_Sc_NS_Change_Of_NodeAddress – This MSC has been removed from the collection due to lack of compliance with the MOST Specification. A NetworkSlave is not allowed to change its NodeAddress during runtime. The NetworkMaster would signal a transition to NotOK in such an error case, as soon as the inconsistency is noticed.
1V3_013	3.5.1.1	NS_Sc_StartupOK – Renamed NetOn_event to InitReady event. NetOn event is no longer used.
1V3_014	3.5.1.2	NS_Sc_StartupNotOK – Renamed NetOn_event to InitReady event. NetOn event is no longer used.
1V3_015	4.4.1.1	CM_Gen_M_CleanUp – Changed guarding condition "when (SourceType = Allocate)" to "otherwise".
1V3_016	4.7	– Added new MSC CM_Boundary_Change.
1V3_017	5.5	PM_Gen_Device_WakeUp – Modeled pre-condition as guarding condition.
1V3_018	5.6	PM_Gen_Overtemp_Shutdown: – Changed AbilityToWake to PermissionToWake. – Switching light off when theta_dead is reached is modeled as exception instead of a mere parallel action.
1V3_019	5.7	PM_Gen_Restart_After_Overtemp_Shutdown – Changed AbilityToWake to PermissionToWake. – In those cases where no corresponding events are modeled, added comments, stating that restarting the network is performed by the NetworkMaster. – Added Over-Temperature-Shutdown broadcast message from device that is still in the overtemperature state. – The device that initiated the over temperature shutdown is allowed to wake up the network. – The PowerMaster may restart the network but is not required to do so. – The network restart may be triggered by the user after $t_{WaitAfterOverTemp}$ has expired.
1V3_020	-	NM_Sc_Avoiding_InstID_Collision – This empty MSC was removed from the collection.
1V3_021	-	NM_Gen_ScanType – This outdated MSC was removed from the collection.

Changes MOST Dynamic Specification Rev 1V2 (04/2006) to MOST Dynamic Specification Rev 1V2 (06/2006)

Change Ref.	Section	Changes
1V2_06_001	3.3.4.12	Substituted unguarded OPT inline expression with additional branch in ALT inline expression to make the behavior deterministic.

Changes MOST Dynamic Specification Rev 1V1 to MOST Dynamic Specification Rev 1V2

Change Ref.	Section	Changes
1V2_001	General	Changed light to modulated signal.
1V2_002	Document References	Added Function Blocks that also affects the Dynamic Specification.
1V2_003	3.2.3	Updated MSC comments and added timer t_WaitBeforeScan.
1V2_004	3.3.2	Changed order in MSC. Added timer t_WaitBeforeScan and action.
1V2_005	3.3.3.2, 3.3.3.3, 3.3.3.4	Added timer t_WaitBeforeScan.
1V2_006	3.3.4.16	Corrected typo in MSC.
1V2_007	3.4.4	Changed order in MSC and added remark.
1V2_008	3.5.1	Deleted section “Startup – Timeout” due to deletion of t_CfgStatus.
1V2_009	4.3.1.2, 4.3.1.5, 4.4.2.1-4.4.2.4, 4.9.3	Added remark that the source activity is optional.
1V2_010	5.3	Added timer t_SlaveShutdown.
1V2_011	4.4.2	Added Chapter and MSCs handling source and sink drop.
1V2_012	7	Updated Timers table.
1V2_013	3.3.4.3, 3.3.4.9	Deleted specific description of reasons for scan initiation.
1V2_014	3.4.2, 3.5.1.1, 3.5.1.2	Replaced $t_{CfgStatus}$ with t_{Answer}. $t_{CfgStatus}$ equivalent removed from MSCs.
1V2_015	5.4	MSC changed with respect to t_RetryShutdown timer.
1V2_016	General	Unified spelling of NetworkMaster and ConnectionMaster.
1V2_017	3.2.3, 3.3.2, 3.3.3.2, 3.3.3.3, 3.3.3.4	Condition NotOK now set before timer t_WaitBeforeScan starts.
1V2_018	3.2.3	OPT inline expression with “Wait until NCE has not occurred…” action removed - already covered by $t_{WaitBeforeScan}$ timer.
1V2_019	3.3.4.6, 3.3.4.8	Configuration.Status(invalid) now directly after Central Registry change detected, “No errors occurred during…” text block removed.
1V2_020	3.3.4.4, 3.3.4.9, 3.3.4.10, 3.3.4.12	Added Remark: “This scenario is only valid for the mechanism of parallel scanning of the system. It does not cover sequential scanning.”
1V2_021	3.4.4	Deleted Configuration.Status(NotOK) message - already contained in referencing MSCs.
1V2_022	5.2	MSC change: Removed the idle loop. Removed the alternative path that deals with devices that do not have the permission to wake the network.
1V2_023	3.3.4.2	MSC change: Configuration.Status(Invalid) is now sent immediately after a conflict occurs. At which point Configuration.Status(New) is sent now depends on whether the NetworkMaster supports “immediate notification”.
1V2_024	3.3.4.3	SystemState(NotOK) event added as trigger for this MSC. MSC change: At which point Configuration.Status(New) is sent now depends on whether the NetworkMaster supports “immediate notification”.
1V2_025	3.3.4.11	Original section 3.3.4.12 NM_Sc_NCE_SystemState_To_OK has been split into two; NM_Sc_NCE_SystemStateNotOK_To_OK now only deals with SystemState NotOK.
1V2_026	3.3.4.12	New section, derived from former section 3.3.4.12 NM_Sc_NCE_SystemState_To_OK. Here, the focus is on SystemState OK. The MSC now differentiates between NetworkMasters with or without the “immediate notification” feature.

Change Ref.	Section	Changes
1V2_027	3.3.3.3, 3.3.4.5, 3.3.4.6, 3.3.4.7, 3.3.4.8	Corrected inconsistent use of timer t_DelayCfgRequest: Timers t_DelayCfgRequest1 and t_DelayCfgRequest2 were renamed to t_DelayCfgRequest. The latter is now initialized with values t_DelayCfgRequest1 and t_DelayCfgRequest2, in accordance with 3.2.5 NM_Gen_ReceiveConfiguration. Changed guarding condition “node has been scanned less than 20 times without answering” to “ScansWithoutAnswer < 20” and added improved description as comment.

Changes MOST Dynamic Specification Rev. 1V0 to MOST Dynamic Specification Rev 1V1

Change Ref.	Section	Changes
1V1_001	General	Deleted old chapters 3.1.1 and 3.1.2.
1V1_002	3	Changed un-initialized logical node address from 0x0FFD to 0xFFFF.
1V1_003	3.3.2	Changed order in MSC.
1V1_004	3.3.4.12	MSC23 compliant with MSC5.
1V1_005	4.3	Added chapter.
1V1_006	4.4	Timeout replaced abort in connection management.
1V1_007	4.4.1.2.1	Sink changed to source.
1V1_008	4.5	Timeout replaced abort in connection management.
1V1_009	5	Added chapter.

Bibliography

All documents, which are referenced by this MOST document, are listed here along with their versions.

Number	Document	Revision
[1]	MOST Specification	3.0
[2]	MOST FBlock NetworkMaster	3.0.2
[3]	MOST FBlock ConnectionMaster	3.0.2
[4]	MOST FBlock NetBlock	3.0.2
[5]	MOST FBlock Template GeneralFBlock	3.0.4

1 Introduction

1.1 Purpose

The MOST Dynamic Specification is complementary to the MOST Specification and the MOST Function Library. The behavior of Controller–Slave (FBlock) communication is described with Message Sequence Charts (MSC).

1.2 Scope

The scope of MSCs in this specification is to describe dynamic communication sequences between Controllers and Slaves. The Dynamic Specification covers the main scenarios.

2 MSC Structuring

There are many different ways to describe a MOST System. In this specification, we look at the system as consisting of Slaves or functional services (FBlocks) which are managed by different Controllers.

When looking at the general behavior in a MOST Network (e.g., network startup procedures), all NetworkSlaves are managed by the NetworkMaster.

The Connection Management manages all connections between Slaves.

Audio and video implemented in larger systems are managed by logical entities, Audio or Video Management. The purpose of these is to control amplifiers and displays. In some systems, these are implemented as a part of the HMI but could also be implemented as separate devices. Therefore, Synchronous Management (Audio, Video, or Camera Management) is useful in order to keep complex systems well structured.

After the startup of a MOST system, it is possible for different Controllers to act simultaneously. Timers and other constraints may affect this behavior. Communication in the MOST System could, therefore, be seen as consisting of many Controller–Slave communication sequences which are either mandatory or optional.

2.1 General MSCs vs. Scenario MSCs

The purpose of the general MSCs is to describe—as completely as possible—the dynamic behavior of the different functional areas (e.g., NetworkMaster, NetworkSlave, or Connection Management). All relevant events and responses from the communication partner are considered. The high-level MSC of a specific functional area shows how the general MSCs are combined to describe the complete behavior of this area.

The purpose of the scenario MSCs is to extract a specific path from the general MSCs and thereby show a simple case. To reduce the total number of MSCs, there can still be alternative or optional paths inside the scenario MSCs (e.g., handle different responses to a sent message). However, the intention is to describe the different example cases as simple as possible.

3 Network Management

The MSCs in this section show how the NetworkMaster maintains the Central Registry by collecting configuration information from all NetworkSlaves. The NetworkMaster then distributes information about the System State to all NetworkSlaves.

3.1 NetworkMaster General MSCs

The general NetworkMaster MSCs are divided into two parallel processes. One process requests configurations from the NetworkSlaves when required. The other process receives the registrations when provided by the NetworkSlaves. The latter process checks the validity of the registrations. All NetworkSlaves in the network are treated individually.

3.1.1 Variables used in general NetworkMaster MSCs

The general MSCs use variables to simplify the MSCs, as well as reduce the total number of MSCs. Table 3-1 shows a list of the variables used in the general NetworkMaster MSCs. Figure 3-1 shows an example of what a Central Registry using some of these variables may look like.

Note that these variables and the Central Registry in figure Table 3-1 and Figure 3-1, respectively, are used only to show the behavior and do not specify the actual implementation of the NetworkMaster.

Variable	Range	Explanation
numErr_nodepos[1]	0...2	The number of times that this node has caused Configuration.Status(NotOK) in succession. If the same node causes Configuration.Status(NotOK) three times in succession, then the node will be ignored until the next NCE or system restart.
request_nodepos[1]	True, False	Holds information if this node should be scanned. If request_4 is set to true, then the node at node position four should be scanned.
$t_{nodepos}$[1]	0...$t_{WaitForAnswer}$	This is a $t_{WaitForAnswer}$ for each node.
numCompScans	0…∞	The number of times the NetworkMaster has made complementary scans. The value of numCompScans affects the time between the complementary scans. If the node has been scanned less than 20 times, then the $t_{DelayCfgRequest1}$ is used, otherwise $t_{DelayCfgRequest2}$ is used instead.
doRequest	True, False	If true, this variable tells the NetworkMaster to scan all nodes that have request_nodepos set to true.
ConfigUpdate	Success, Error	This variable indicates if the last registration was correct. It is used when updating the configuration status of the network and broadcasting the result of a scan or registration.
numNodes	1…64	The number of nodes to scan.
NWM_supports_immediate_notification	True, False	If this variable is true, the NetworkMaster notifies the NetworkSlaves of FBlocks that were added to the Central Registry immediately during a system scan. If it is false, the NetworkMaster waits until the system scan has completed before
NewFBlocksRegistered	True, False	If true, new FBlocks were added to the Central Registry during a system scan.
FBlocksRemoved	True, False	If true, FBlocks were removed from the Central Registry during a system scan.
NoChangesToCentralRegistry	True, False	If true, the Central Registry has not changed during a system scan.

Table 3-1: Variables used in the general Network Management MSCs

Figure 3-1 shows a cleared Central Registry in a network with four nodes before the NetworkMaster starts scanning the network. The example Central Registry uses some of the variables in Table 3-1.

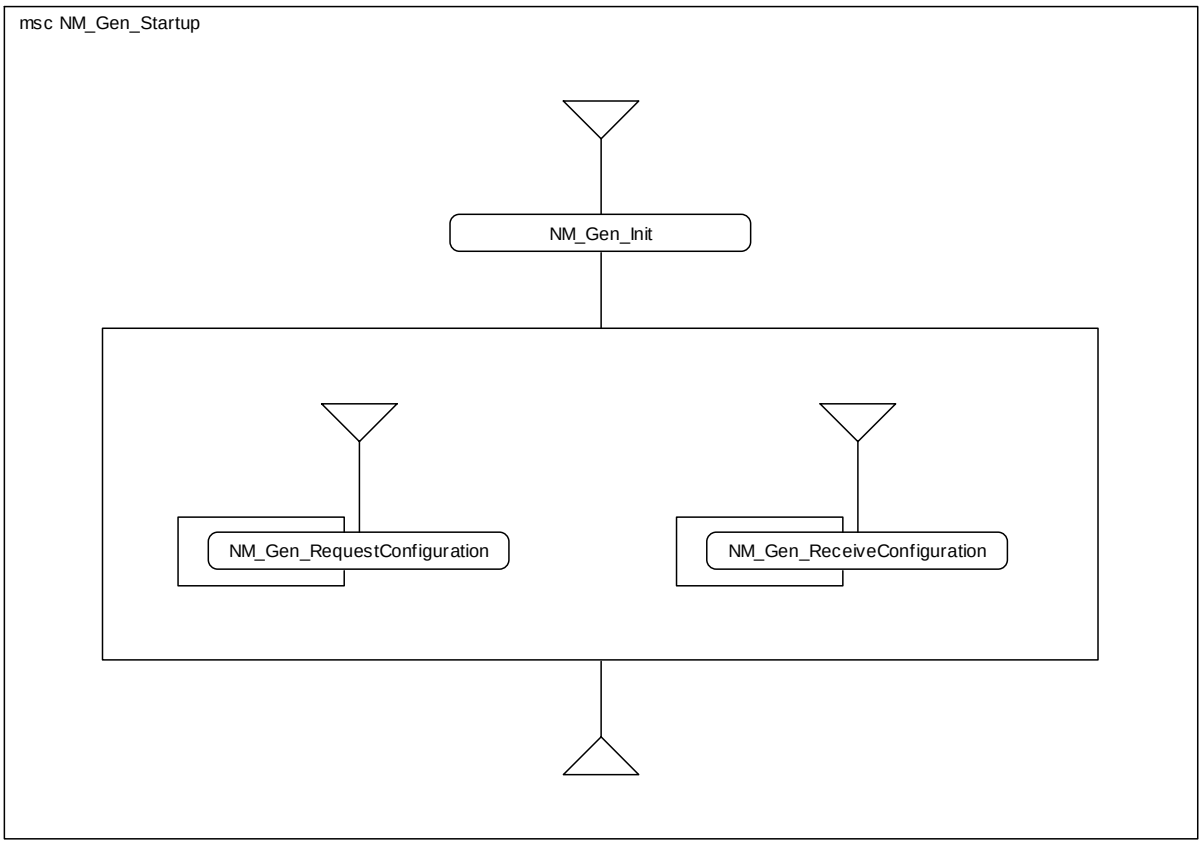
Node position	NodeAddress	FBlockID	InstID	request_nodepos	numErr_nodepos	$t_{nodepos}$
0	-	-	-	request_0 = True	numErr_0 = 0	t_0
1	-	-	-	request_1 = True	numErr_1 = 0	t_1
2	-	-	-	request_2 = True	numErr_2 = 0	t_2
3	-	-	-	request_3 = True	numErr_3 = 0	t_3

Figure 3-1: An example of what a Central Registry may look like.

[1] "nodepos" is replaced by the actual node position of the node.

3.1.2 High-level NetworkMaster MSC

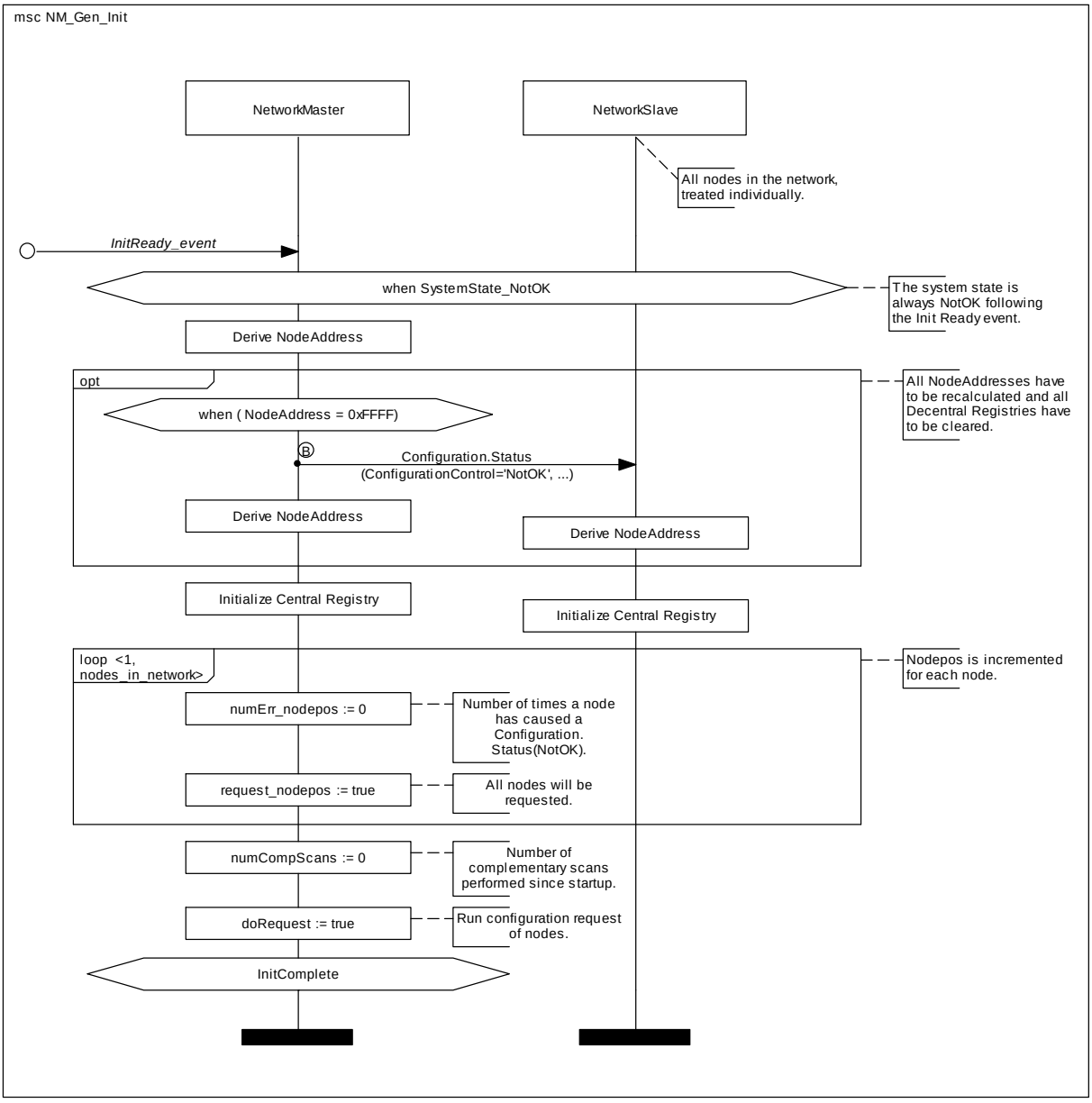
General MSC:	NM_Gen_Startup
Description:	High-level MSC of NetworkMaster network configuration process. After detecting the Init Ready event, the NetworkMaster initializes itself; this is shown in NM_Gen_Init. When the NM_Gen_Init has completed, two parallel processes are started. One process (NM_Gen_RequestConfiguration) asks nodes for their configuration and one process (NM_Gen_ReceiveConfiguration) handles the reception of registration messages. These two processes run in parallel until shutdown.
Prior Condition:	–
Initiator:	–
Events	Init Ready
Timers/Timing constraints	–
Remarks:	–



MSC 1: NM_Gen_Startup

3.1.3 Initializing the NetworkMaster

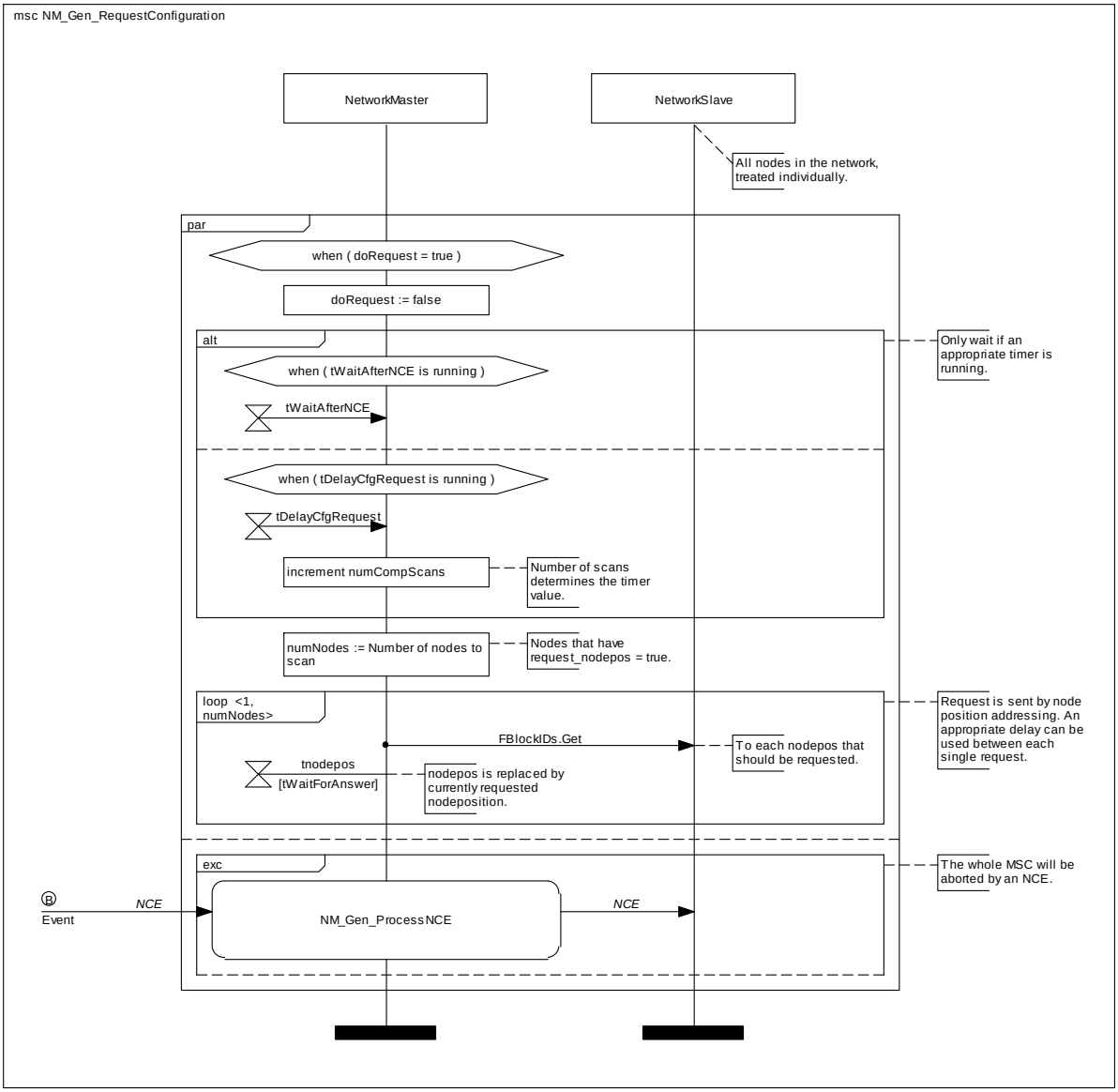
General MSC:	NM_Gen_Init
Description:	The NetworkMaster initializes its node address and resets all variables used during scanning. It also sets “request_nodepos” for all nodes; this leads to all nodes being scanned, as well as setting “doRequest” which triggers the scanning process.
Prior Condition:	NetInterface Init
Initiator:	–
Events	Init Ready
Timers/Timing constraints	$t_{WaitBeforeScan}$
Remarks:	–



MSC 2: NM_Gen_Init

3.1.4 Requesting Configuration

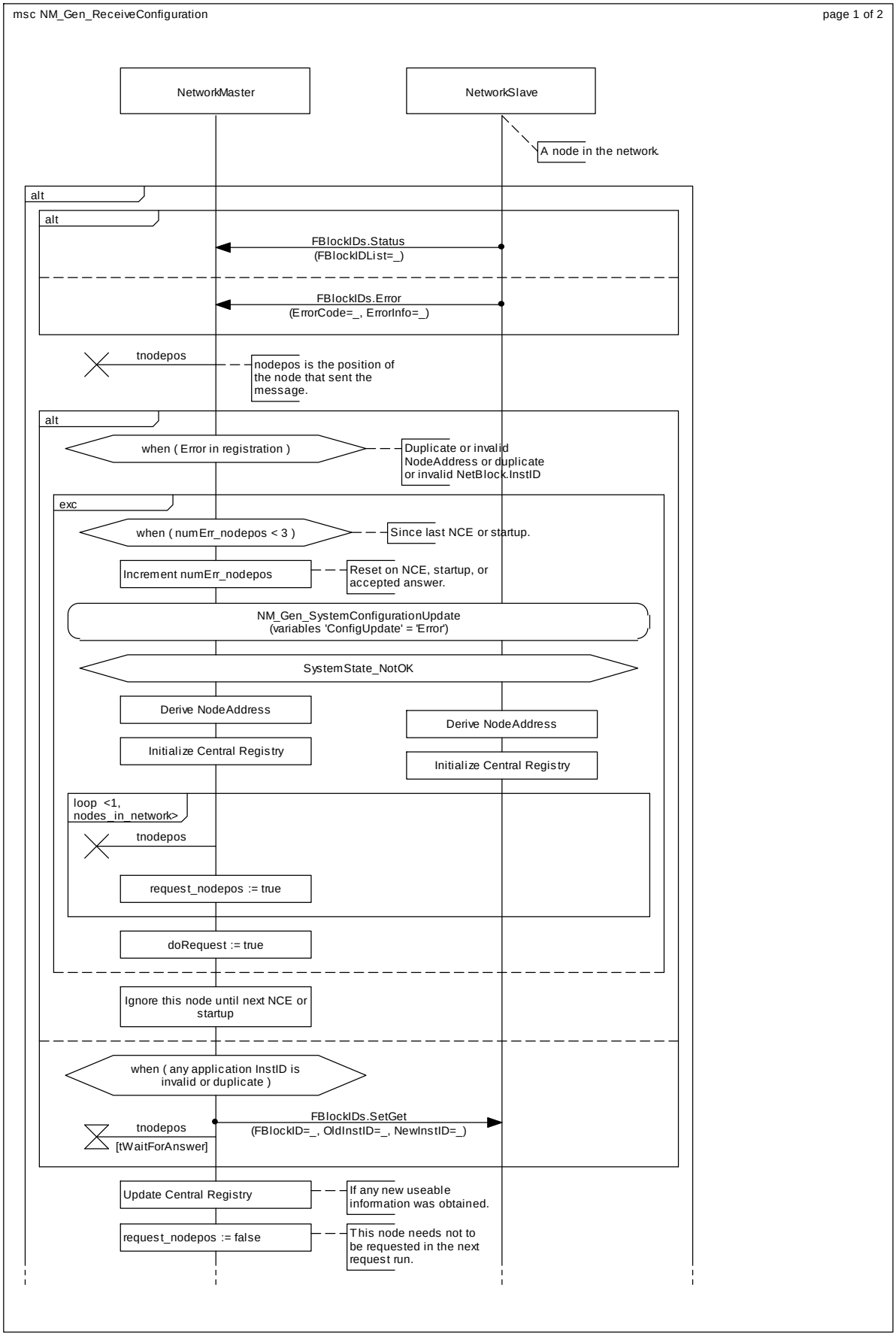
General MSC:	NM_Gen_RequestConfiguration
Description:	This process requests the configuration from nodes that have their “request_nodepos” set.
Prior Condition:	doRequest = True
Initiator:	NetworkMaster
Events	NCE
Timers/Timing constraints	– $t_{DelayCfgRequest}$ – $t_{WaitAfterNCE}$ – $t_{WaitForAnswer}$
Remarks:	– An NCE interrupts this process. – $t_{DelayCfgRequest}$ and $t_{WaitAfterNCE}$ never run simultaneously. Please refer to section 3.1.7.

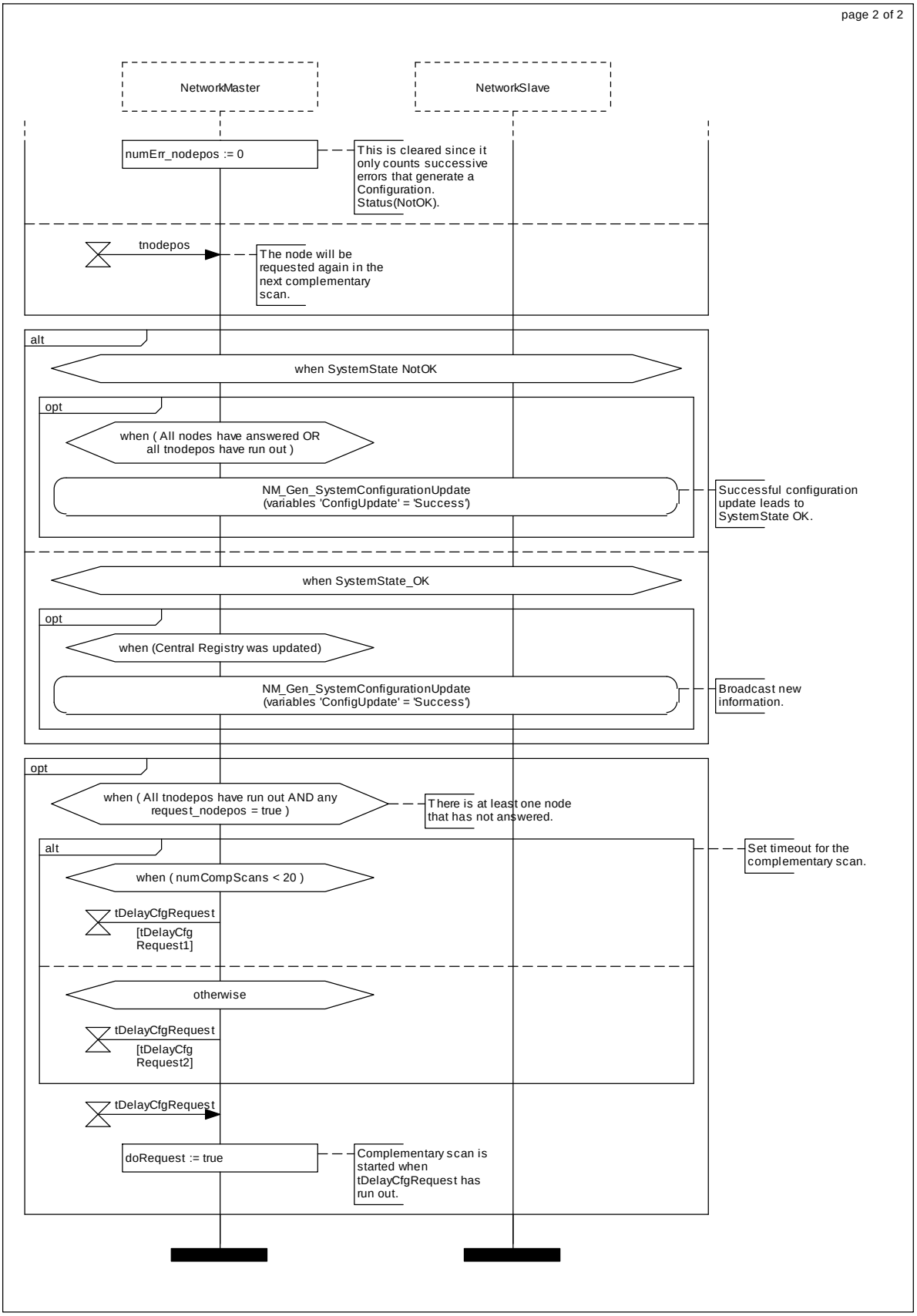


MSC 3: NM_Gen_RequestConfiguration

3.1.5 Receiving Registrations

General MSC:	NM_Gen_ReceiveConfiguration
Description:	This process handles the reception of registration messages from the NetworkSlaves.
Prior Condition:	Normal Operation
Initiator:	Any NetworkSlave
Events	–
Timers/Timing constraints	– $t_{WaitForAnswer}$ – $t_{DelayCfgRequest}$
Remarks:	In this scenario, an NCE does not occur.

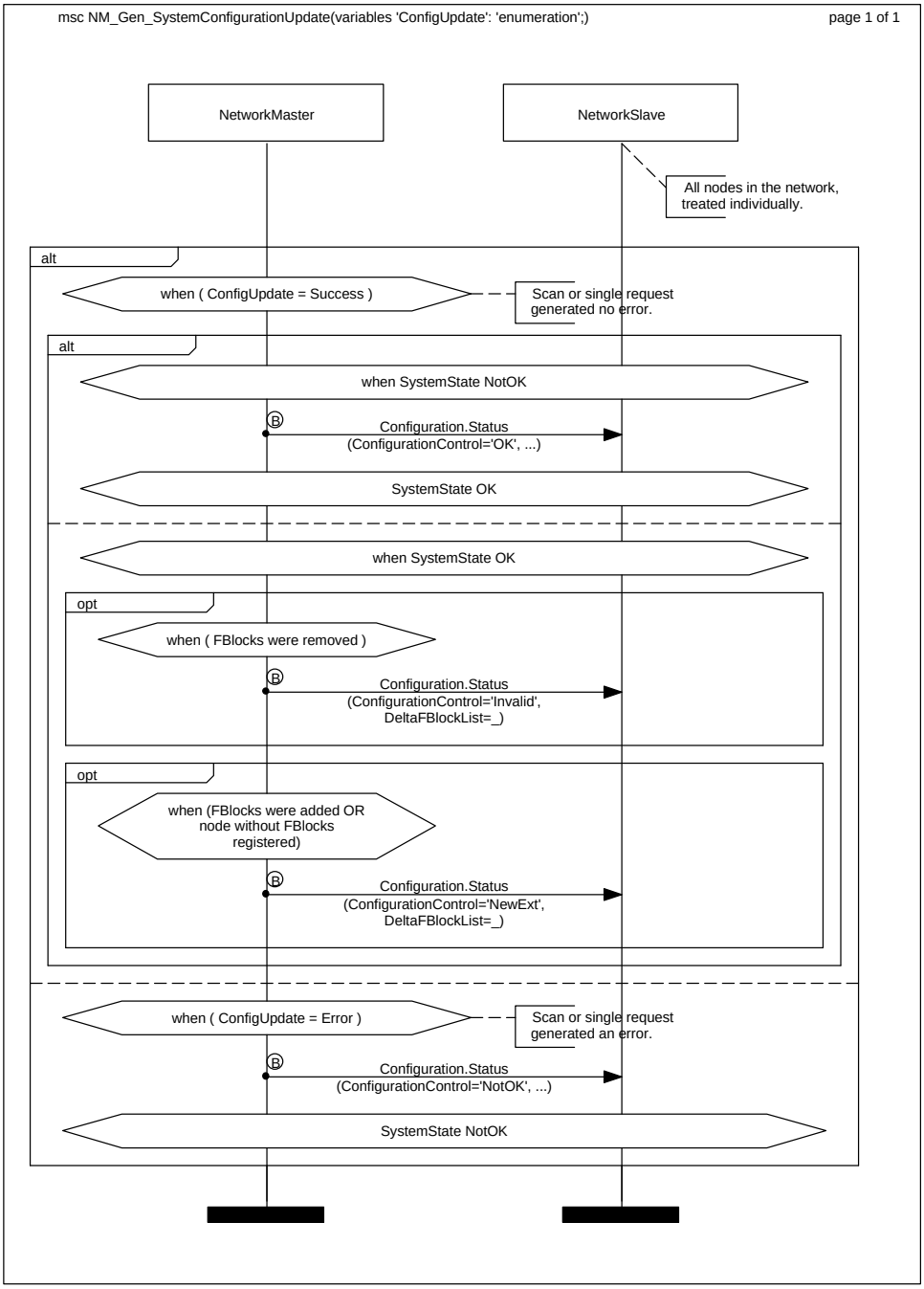




MSC 4: NM_Gen_ReceiveConfiguration

3.1.6 Updating System State

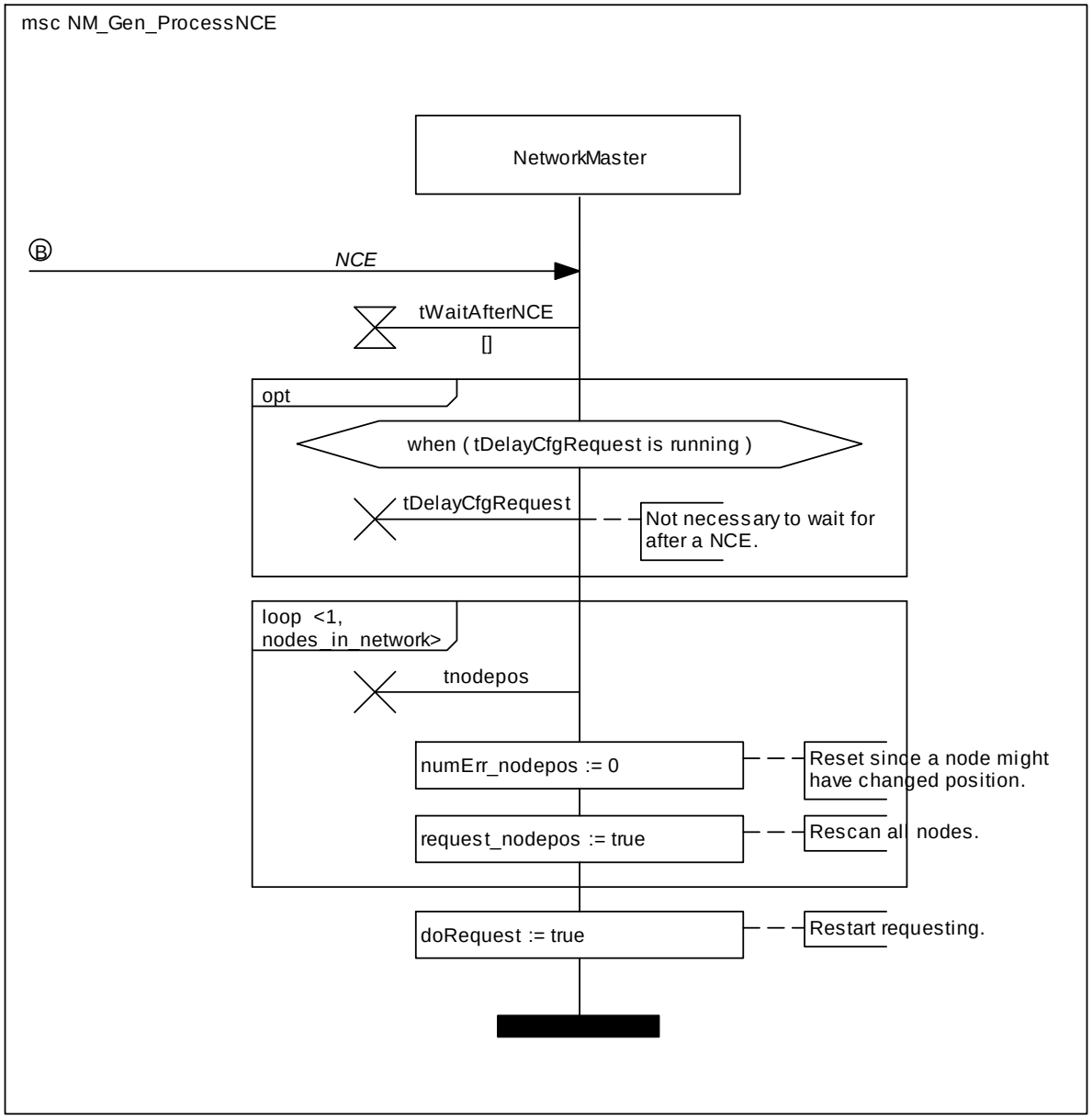
General MSC:	NM_Gen_SystemConfigurationUpdate
Description:	This MSC shows how the NetworkMaster determines the value of the ConfigurationControl parameter of the Configuration.Status() message. The value of the ConfigurationControl parameter depends on the current System State and the value of ConfigUpdate.
Prior Condition:	–
Initiator:	–
Events	–
Timers/Timing constraints	–
Remarks:	–



MSC 5: NM_Gen_SystemConfigurationUpdate

3.1.7 Processing NCEs

General MSC:	NM_Gen_ProcessNCE
Description:	When an NCE is detected, the NetworkMaster interrupts its action and scans the network. This MSC shows how the NetworkMaster resets and sets the relevant properties.
Prior Condition:	–
Initiator:	Any node switching its bypass.
Events	NCE
Timers/Timing constraints	– $t_{DelayCfgRequest}$ – $t_{WaitAfterNCE}$ – $t_{WaitForAnswer}$ (t_nodepos)
Remarks:	–



MSC 6: NM_Gen_ProcessNCE

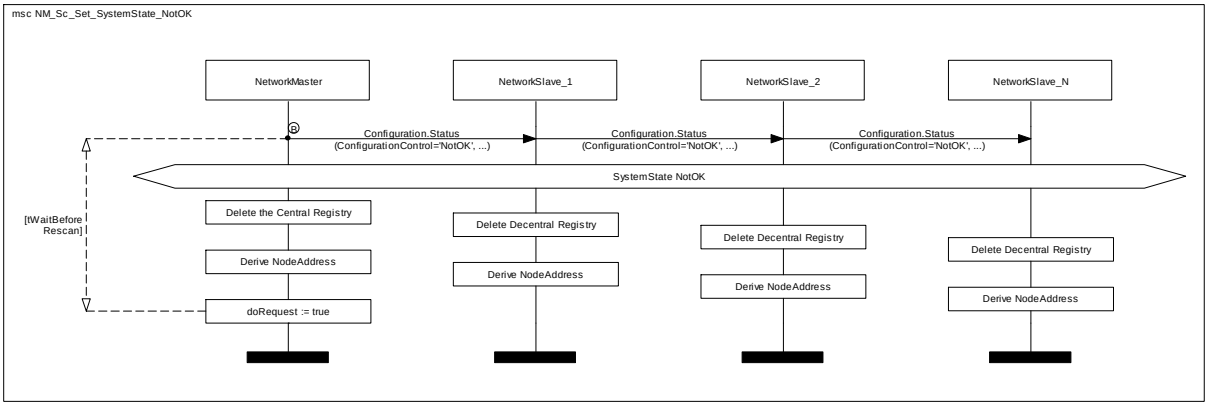
3.2 NetworkMaster Scenario MSCs

This section contains a selection of scenarios that describe the basic behavior of the NetworkMaster.

3.2.1 Setting the System State to NotOK

This scenario is used in the other scenarios whenever the system is reset from a network point of view.

Scenario MSC:	NM_Sc_Set_SystemState_NotOK
Description:	This scenario shows what happens in the network when the NetworkMaster broadcasts Configuration.Status(NotOK). – The Central Registry is cleared. – All Decentral Registries are cleared. – All nodes recalculate their node address. – The System State is set to NotOK.
Prior Condition:	An error occurred during the system scan.
Initiator:	NetworkMaster
Events	–
Timers/Timing constraints	$t_{WaitBeforeScan}$
Remarks:	– This scenario is valid for all System States. – Spontaneous registrations do not occur.



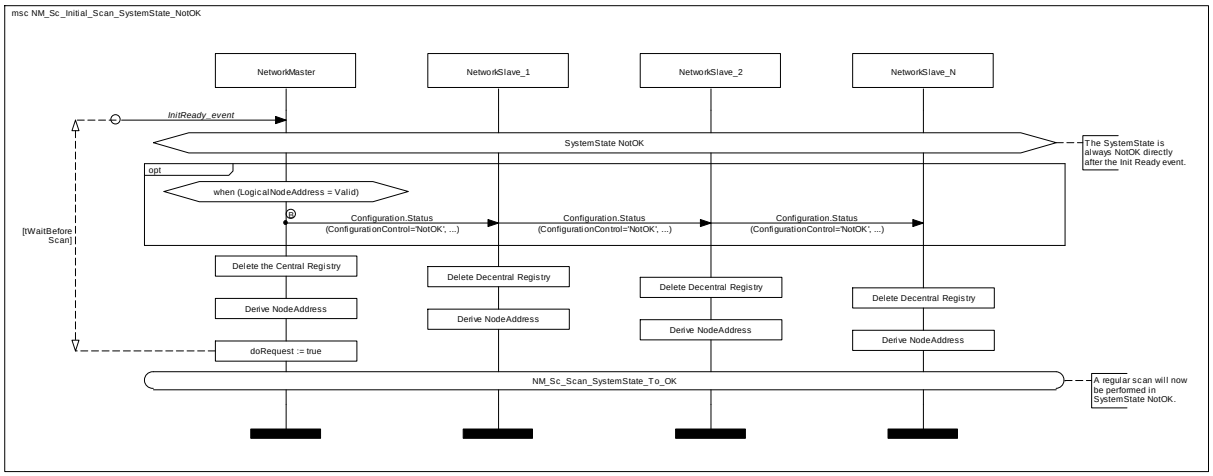
MSC 7: NM_Sc_Set_SystemState_NotOK

3.2.2 Initial Scan

Initial scans follow directly after an Init Ready event until a Configuration.Status() message is transmitted.

3.2.2.1 Initial Scan

Scenario MSC:	NM_Sc_Initial_Scan_ SystemState_NotOK
Description:	This scenario is started by the Init Ready event. When the Init Ready event is detected, the NetworkMaster broadcasts Configuration.Status(NotOK) to re-initialize the system. The NetworkMaster then starts a Regular scan.
Prior Condition:	–
Initiator:	NetworkMaster
Events	Init Ready
Timers/Timing constraints	–
Remarks:	–

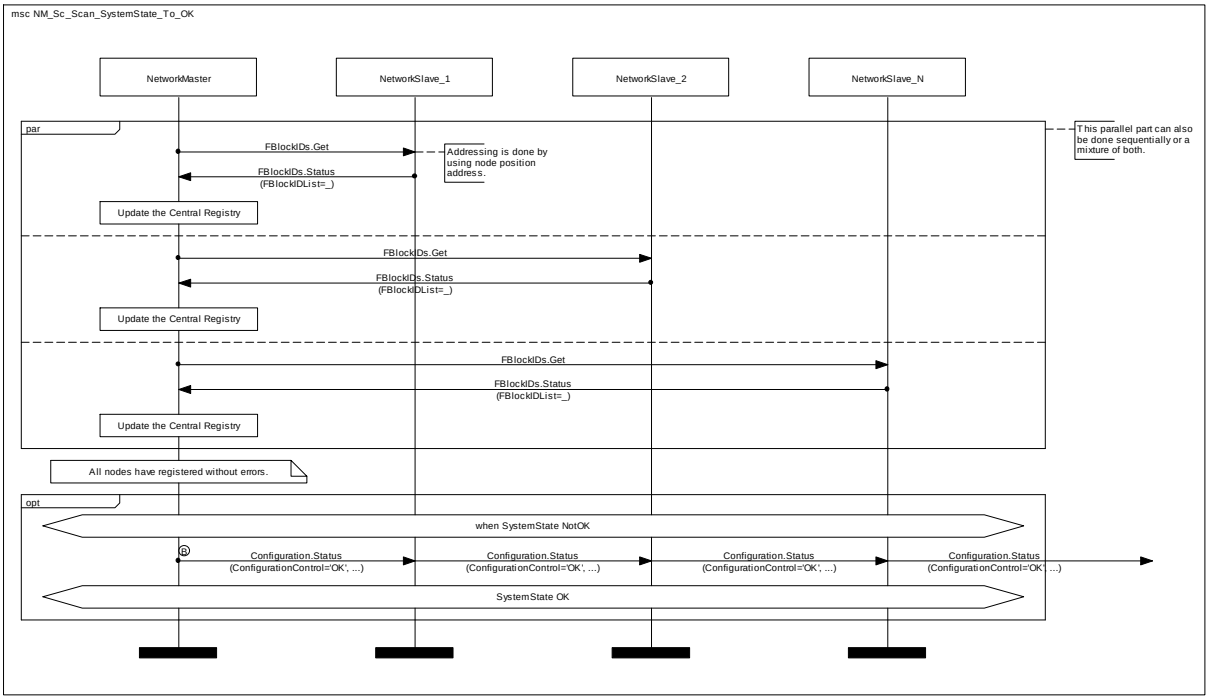


MSC 8: NM_Sc_Initial_Scan_SystemState_NotOK

3.2.3 Regular Scan

3.2.3.1 Normal Scan

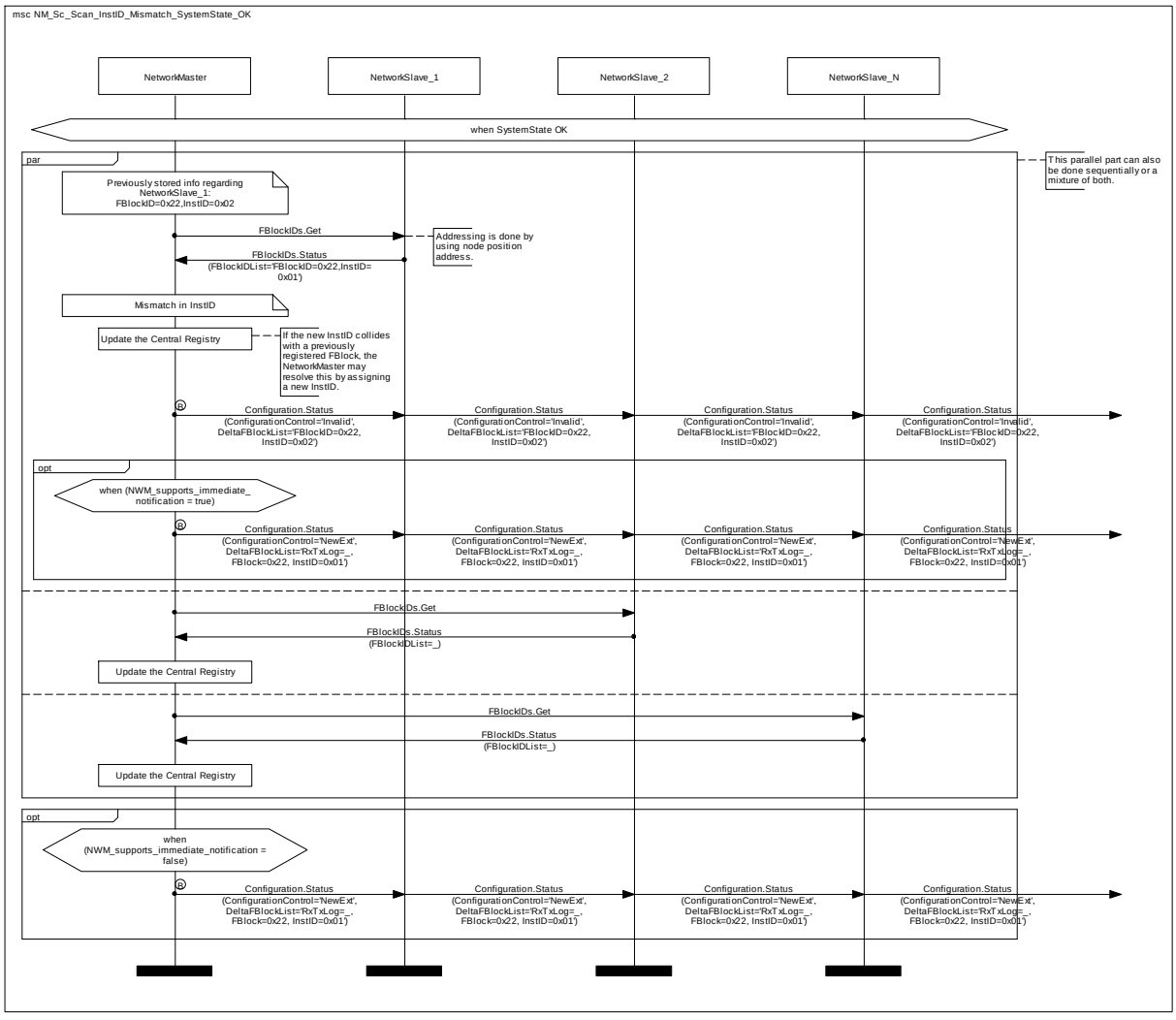
Scenario MSC:	NM_Sc_Scan_SystemState_To_OK
Description:	The NetworkMaster initiates a scan.
Prior Condition:	–
Initiator:	NetworkMaster
Events	Application request (optional) or Configuration.Status(NotOK)
Timers/Timing constraints	–
Remarks:	– This scenario is valid for all System States. – All nodes respond correctly and on time.



MSC 9: NM_Sc_Scan_SystemState_To_OK

3.2.3.2 Mismatch in InstID in System State OK

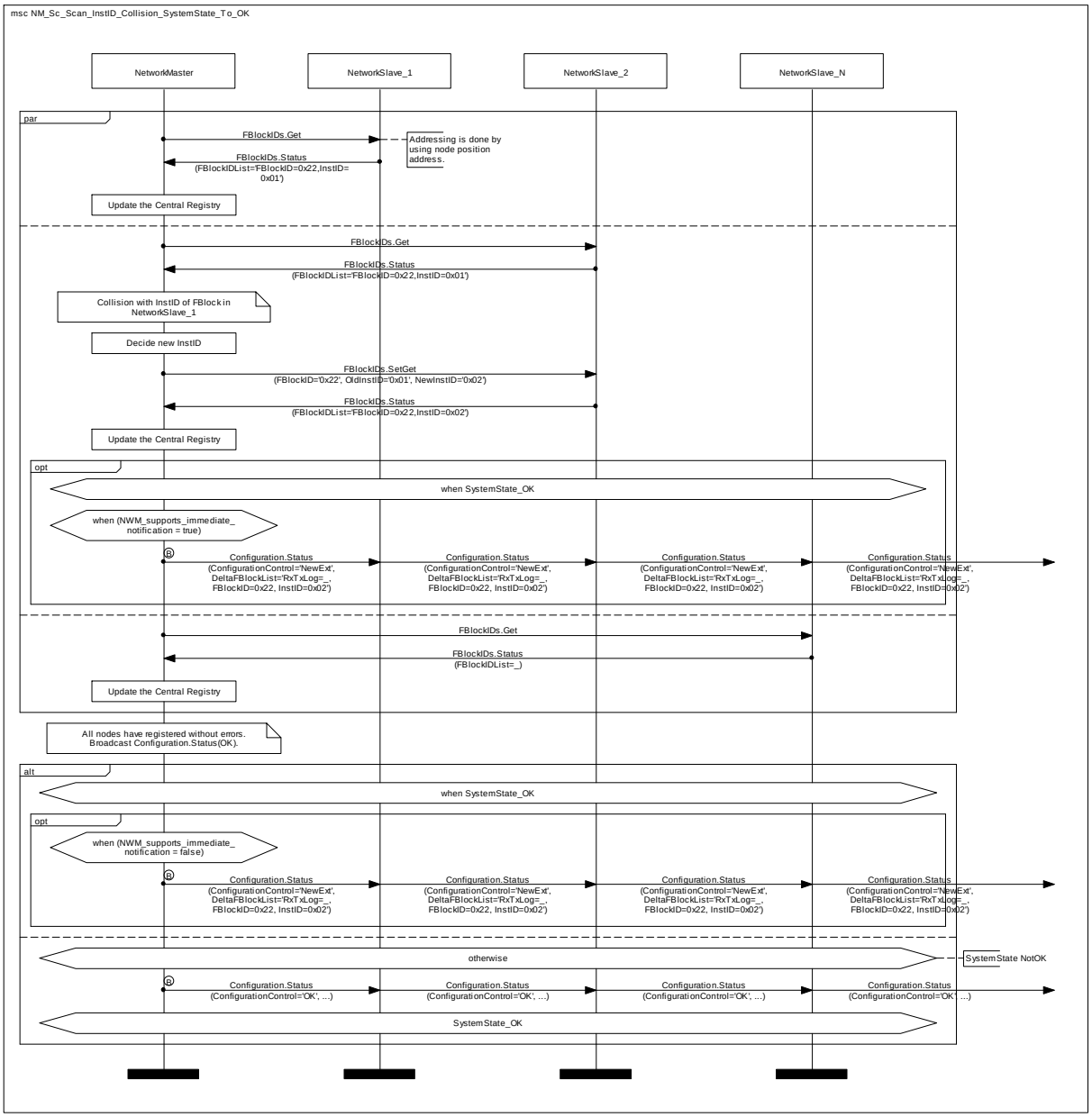
Scenario MSC:	NM_Sc_Scan_InstID_Mismatch_SystemState_OK
Description:	NetworkSlave_1 submits a registration with an InstID mismatch from a previous registration. The NetworkMaster will accept the new registration and broadcast Configuration.Status(Invalid) and Configuration.Status(New). Other nodes register with an FBlockIDList identical to the previous scan.
Prior Condition:	System State OK
Initiator:	NetworkMaster
Events	NCE or an application request (optional)
Timers/Timing constraints	–
Remarks:	– This scenario assumes that the InstID of NetworkSlave_2 does not collide with another FBlock. – This scenario shows the behavior during a scan but the behavior is also applicable on a single node making a registration in System State OK. – The concrete values for FBlockID=0x22 and InstID=0x01/0x02 are examples.



MSC 10: NM_Sc_Scan_InstID_Mismatch_SystemState_OK

3.2.3.3 Collision between InstIDs

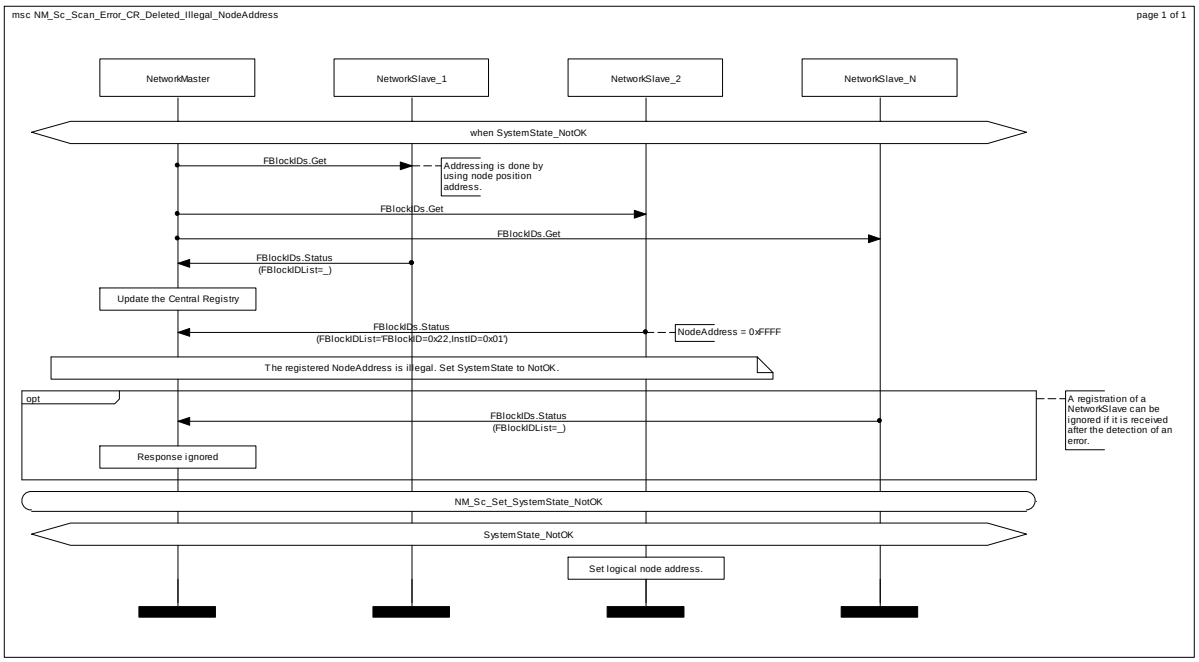
Scenario MSC:	NM_Sc_Scan_InstID_Collision_SystemState_To_OK
Description:	The NetworkMaster scans the network and NetworkSlave_2 registers an FBlock with an InstID that collides with an FBlock instance in NetworkSlave_1. In case there is a collision between two InstIDs, the NetworkMaster can set a new InstID for the colliding FBlock. The NetworkMaster reports the changes to the network differently depending on the current System State.
Prior Condition:	–
Initiator:	NetworkMaster
Events:	NCE, application request (optional), or Configuration.Status(NotOK)
Timers/Timing constraints:	–
Remarks:	– This scenario is valid for all System States. – The concrete values for FBlockID=0x22 and InstID=0x01/0x02 are examples.



MSC 11: NM_Sc_Scan_InstID_Collision_SystemState_To_OK

3.2.3.4 Error when Node Registers an Invalid Node Address

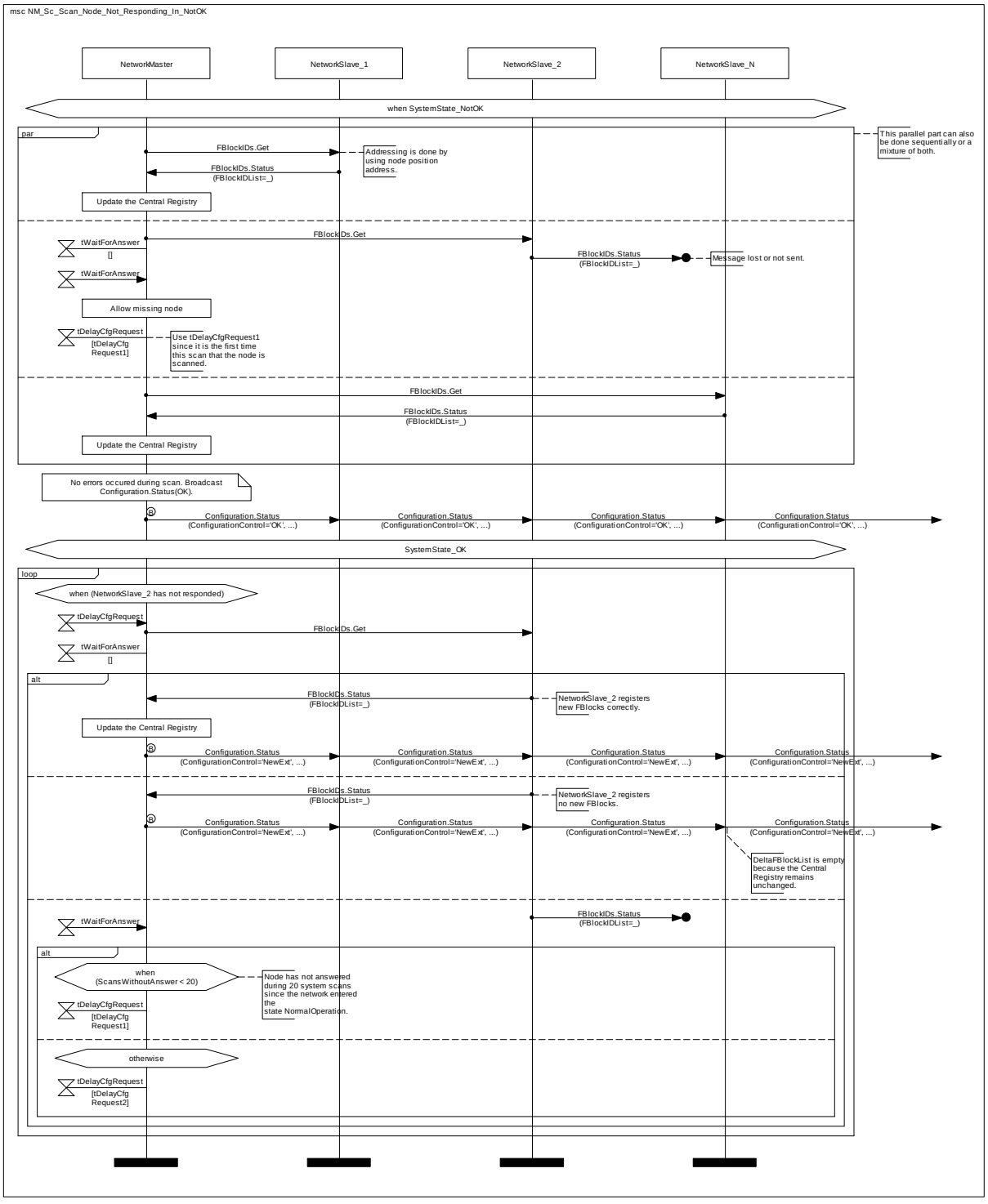
Scenario MSC:	NM_Sc_Scan_Error_CR_Deleted_Illegal_NodeAddress
Description:	In this scenario, NetworkSlave_2 makes a registration with an invalid node address.
Prior Condition:	System State NotOK
Initiator:	NetworkMaster
Events	–
Timers/Timing constraints	–
Remarks:	This scenario is only valid for the mechanism of parallel scanning of the system. It does not cover sequential scanning.



MSC 12: NM_Sc_Scan_Error_CR_Deleted_Illegal_NodeAddress

3.2.3.5 Node Not Responding in System State NotOK

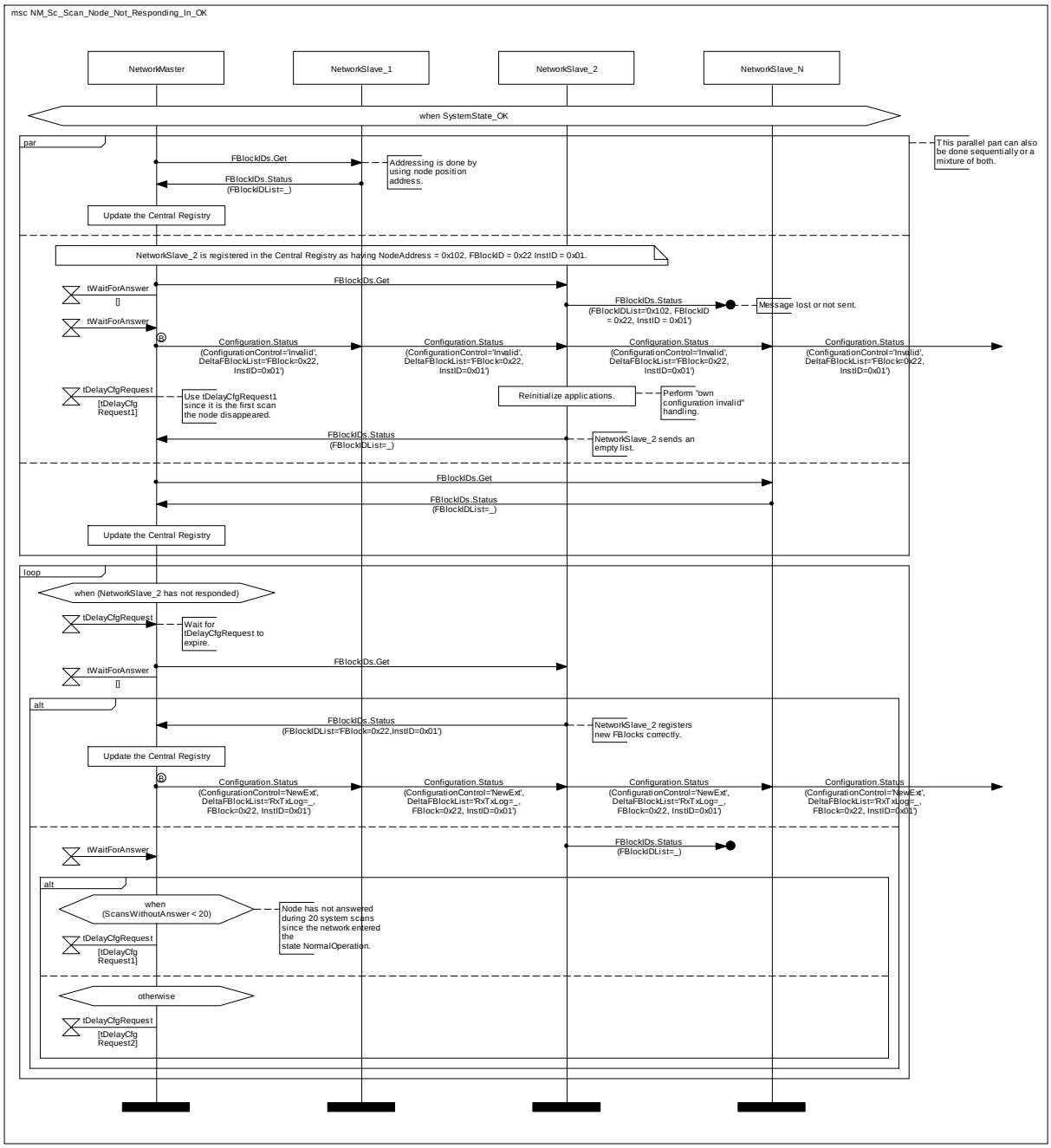
Scenario MSC:	NM_Sc_Scan_Node_Not_Responding_In_NotOK
Description:	The NetworkMaster scans the system in System State NotOK. NetworkSlave_2 does not answer the request in time. The NetworkMaster continues to request its configuration.
Prior Condition:	System State NotOK
Initiator:	NetworkMaster
Events	–
Timers/Timing constraints	– $t_{WaitForAnswer}$ – $t_{DelayCfgRequest}$
Remarks:	See also scenario in section 3.2.3.6.



MSC 13: NM_Sc_Scan_Node_Not_Responding_In_NotOK

3.2.3.6 Node Not Responding in System State OK

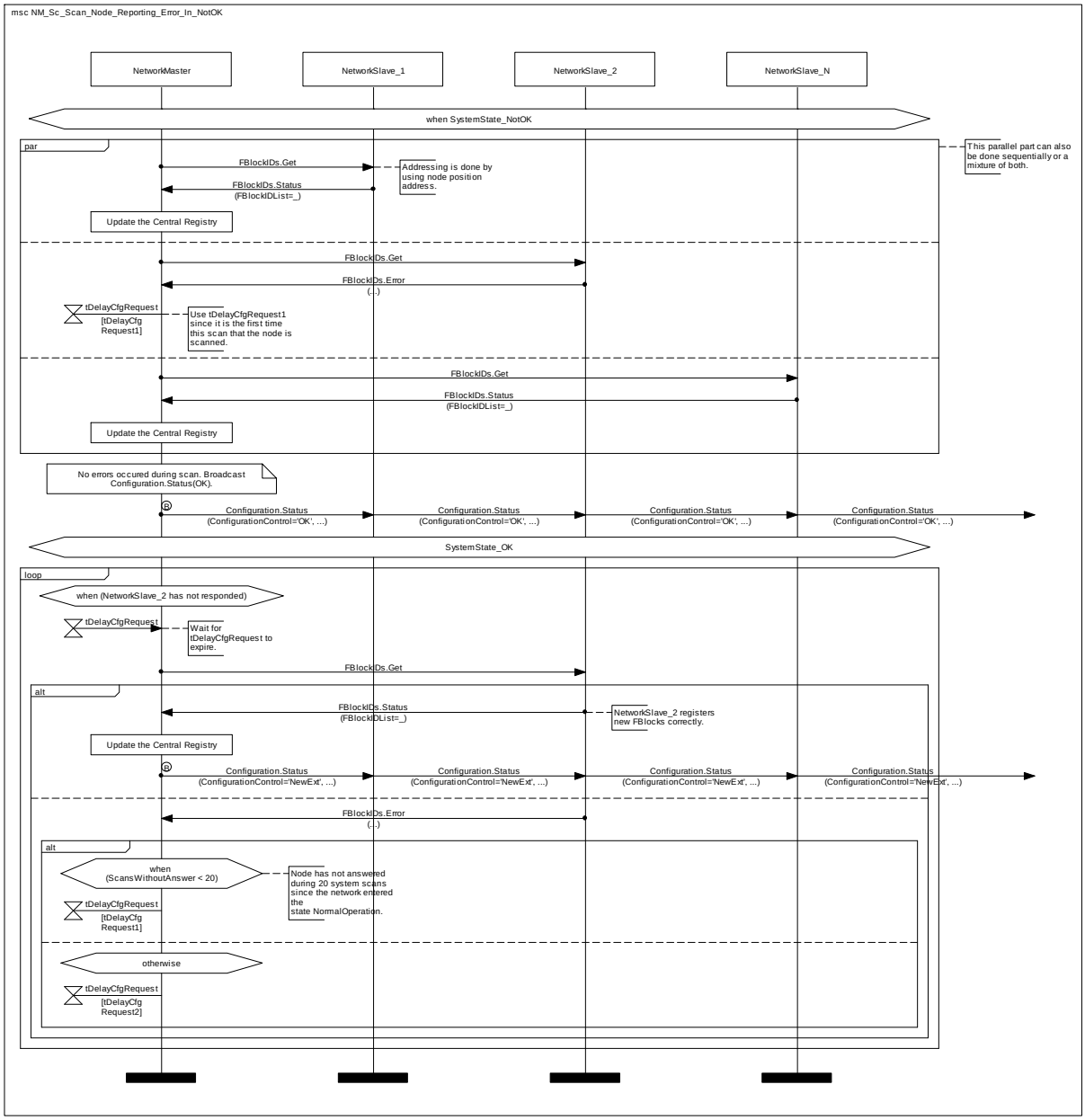
Scenario MSC:	NM_Sc_Scan_Node_Not_Responding_In_OK
Description:	The NetworkMaster scans the system in System State OK. NetworkSlave_2 does not answer the request in time. The NetworkMaster includes this node and continues to request its configuration but it will inform the network of the invalid FBlocks that were previously registered in NetworkSlave_2.
Prior Condition:	System State OK
Initiator:	NetworkMaster
Events	NCE or an application request (optional)
Timers/Timing constraints	– $t_{WaitForAnswer}$ – $t_{DelayCfgRequest}$
Remarks:	See also scenario in section 3.2.3.5.



MSC 14: NM_Sc_Scan_Node_Not_Responding_In_OK

3.2.3.7 Node Reporting Error in System State NotOK

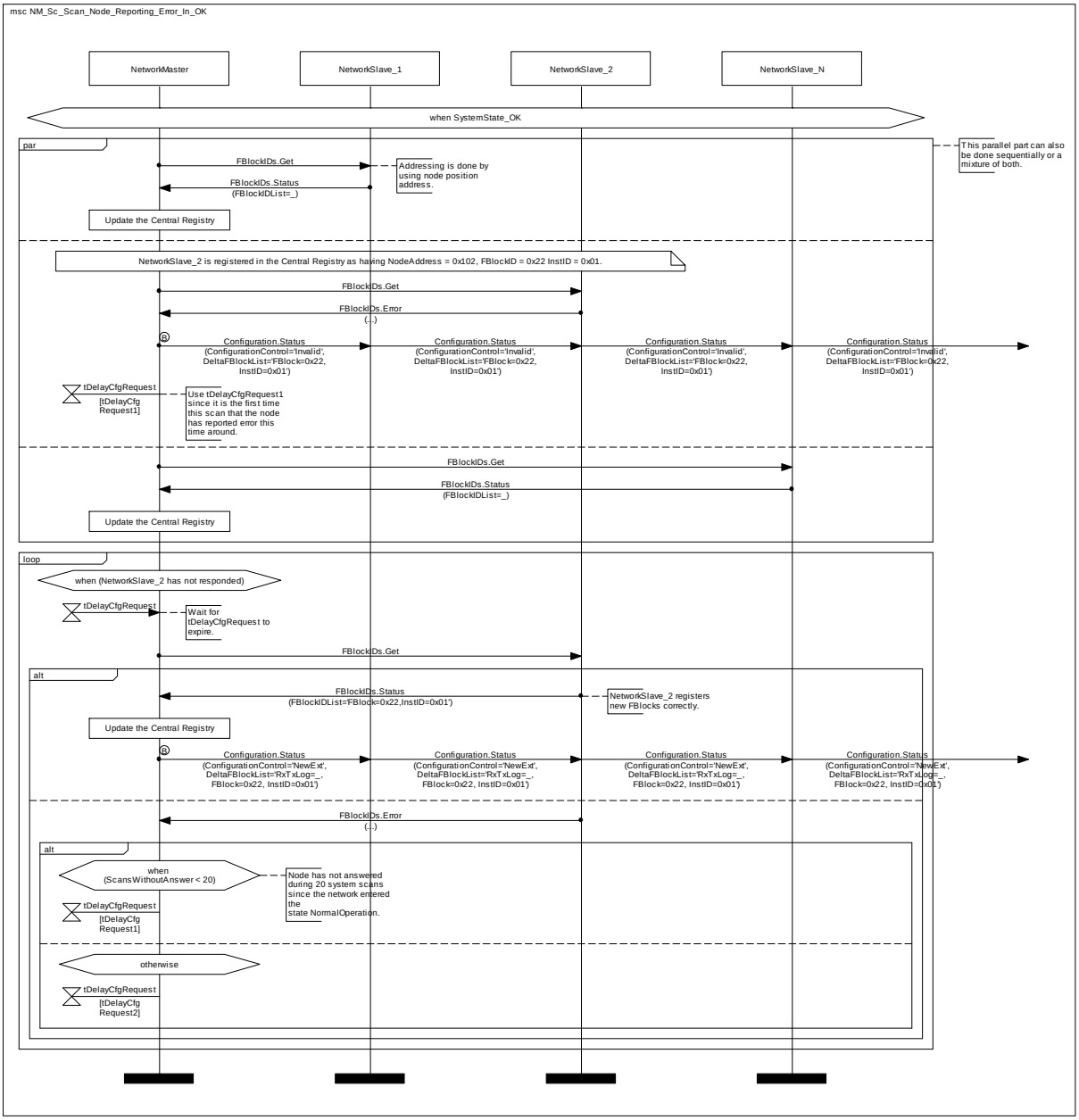
Scenario MSC:	NM_Sc_Scan_Node_Reporting_Error_In_NotOK
Description:	The NetworkMaster scans the system in System State NotOK. NetworkSlave_2 reports Error when the NetworkMaster requests its configuration. The NetworkMaster will treat this node as a non-responding node, and continue requesting FBlockIDs from this node. After 20 retries timer $t_{DelayCfgRequest}$ will change value.
Prior Condition:	System State NotOK
Initiator:	NetworkMaster
Events	–
Timers/Timing constraints	$t_{DelayCfgRequest}$
Remarks:	–



MSC 15: NM_Sc_Scan_Node_Reporting_Error_In_NotOK

3.2.3.8 Node Reporting Error in System State OK

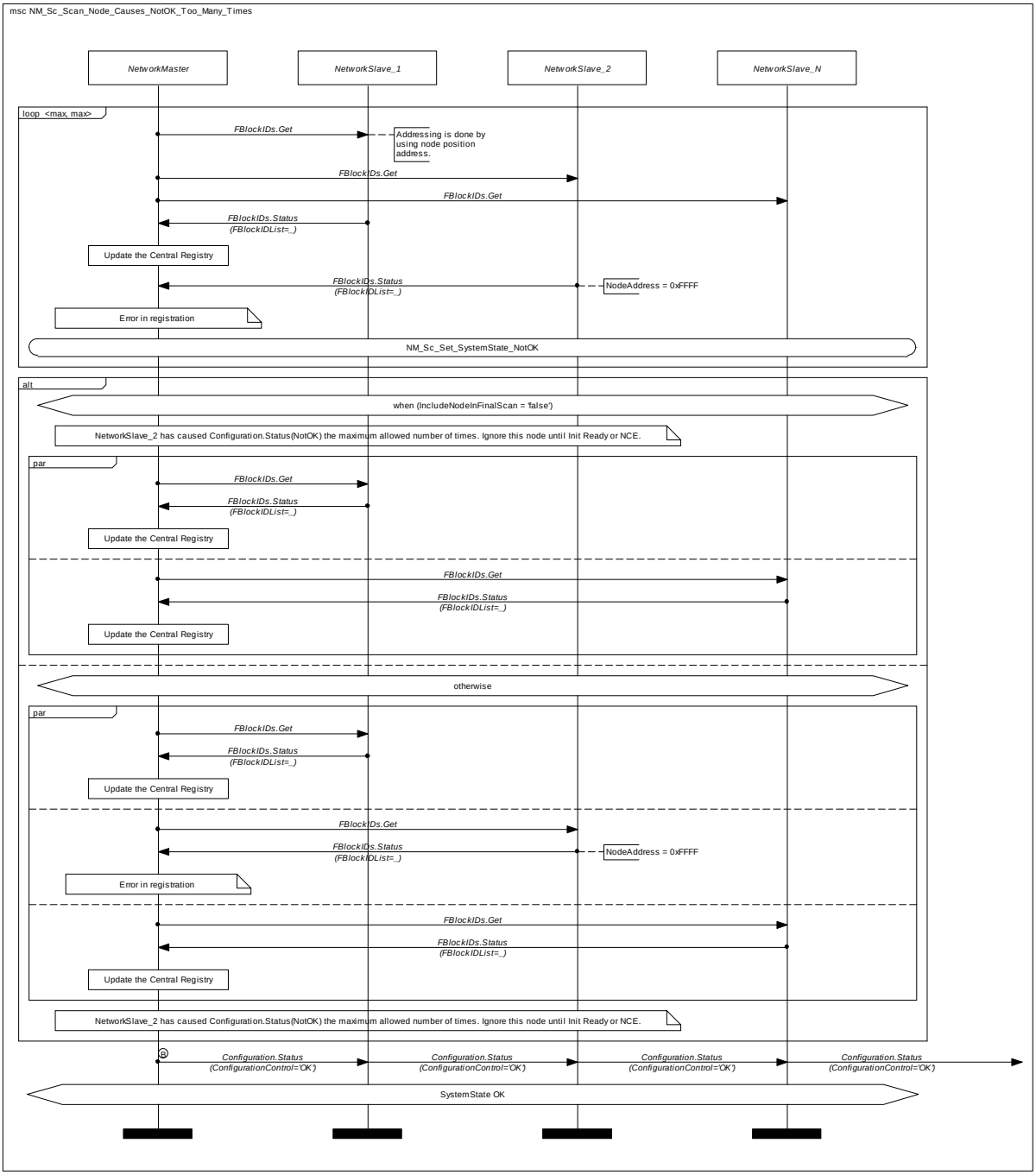
Scenario MSC:	NM_Sc_Scan_Node_Reporting_Error_In_OK
Description:	The NetworkMaster scans the system in System State OK. NetworkSlave_2 reports FBlockIDs.Error() (not ErrorCode 0xA0) when the NetworkMaster requests its configuration. Since NetworkSlave_2 is registered in the Central Registry, the NetworkMaster has to inform the other NetworkSlaves that the FBlock in NetworkSlave_2 is invalid. The NetworkMaster will treat this node as a non-responding node, and continue requesting FBlockIDs from this node. After 20 retries timer $t_{DelayCfgRequest}$ will change value.
Prior Condition:	System State OK
Initiator:	NetworkMaster
Events	NCE or an application request (optional)
Timers/Timing constraints	$t_{DelayCfgRequest}$
Remarks:	–



MSC 16: NM_Sc_Scan_Node_Reporting_Error_In_OK

3.2.3.9 Node causing NotOK too many times

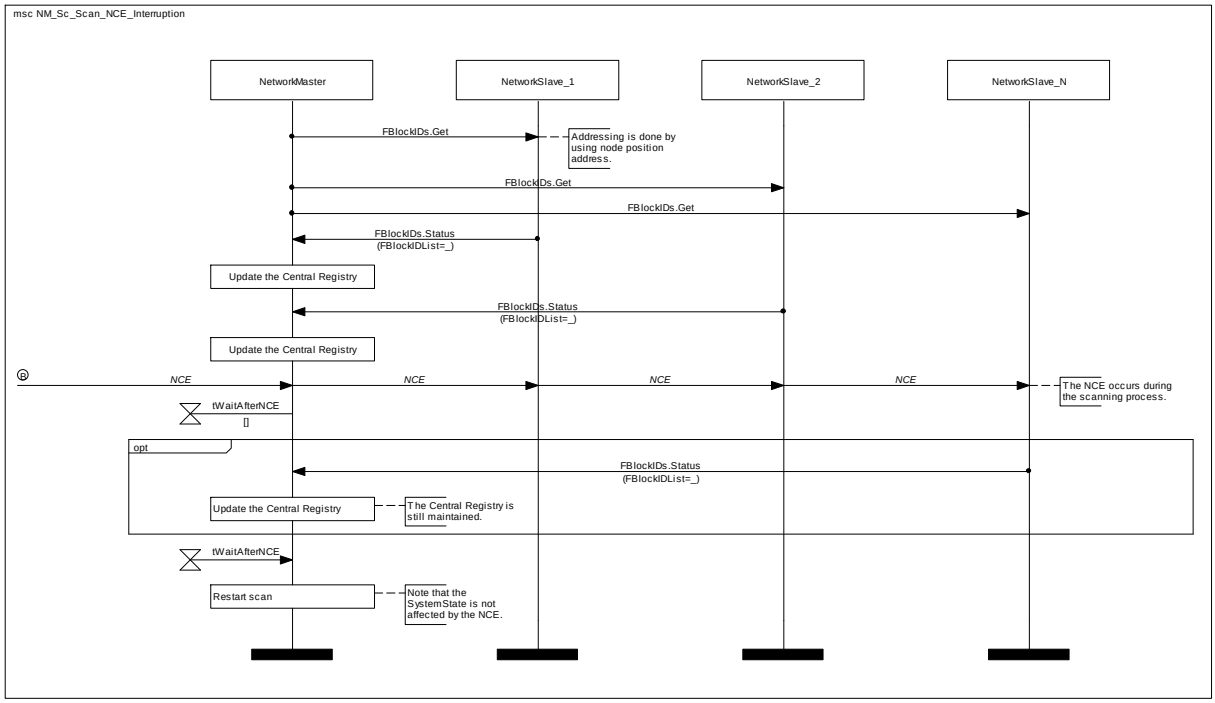
Scenario:	NM_Sc_Scan_Node_Causes_NotOK_Too_Many_Times
Description:	NetworkSlave_2 causes System State NotOK the maximum allowed number of times. The node will be ignored until the next system start or NCE.
Prior Condition:	–
Initiator:	NetworkMaster
Events	NCE
Timers/Timing constraints	–
Remarks:	– This scenario is valid for all System States. – This scenario is only valid for the mechanism of parallel scanning of the system. It does not cover sequential scanning.



MSC 17: NM_Sc_Scan_Node_Causes_NotOK_Too_Many_Times

3.2.3.10 Scan Interrupted by NCE

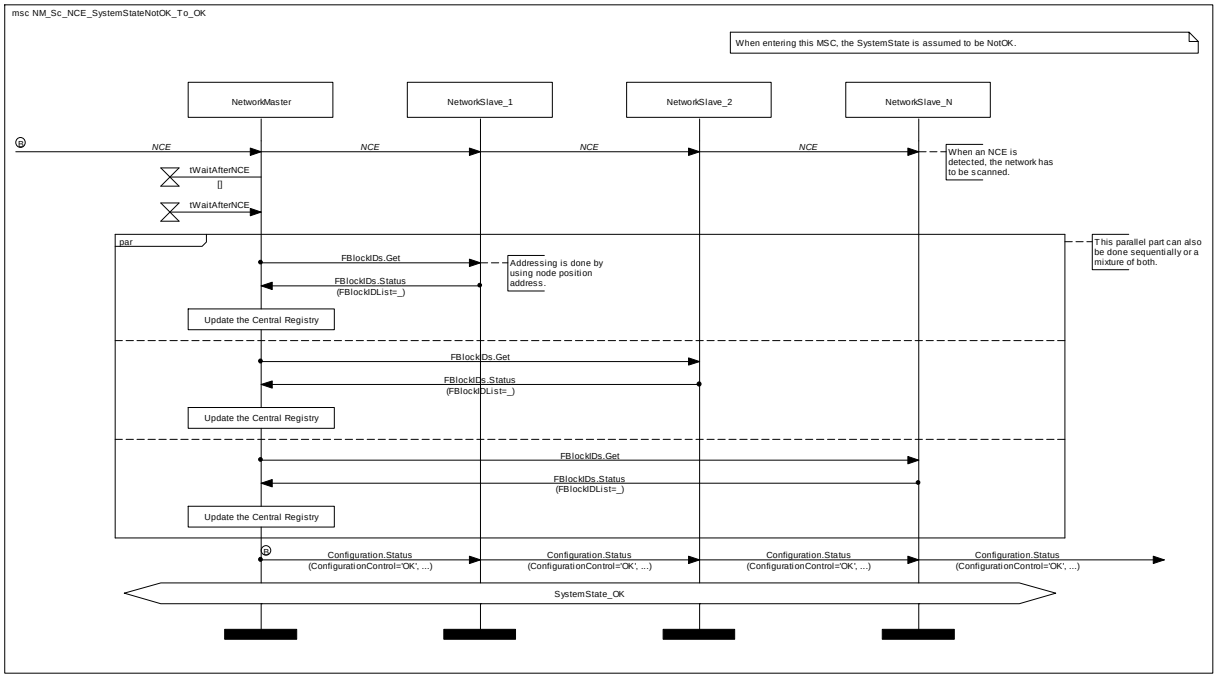
Scenario MSC:	NM_Sc_Scan_NCE_Interruption
Description:	A scan is interrupted by a NCE. Any current scan is restarted when it is interrupted by a NCE regardless of the System State.
Prior Condition:	–
Initiator:	Any node switching its bypass
Events	NCE
Timers/Timing constraints	$t_{WaitAfterNCE}$
Remarks:	– This scenario is valid for all System States – This scenario is only valid for the mechanism of parallel scanning of the system. It does not cover sequential scanning.



MSC 18: NM_Sc_Scan_NCE_Interruption

3.2.3.11 NCE in System State NotOK Resulting in System State OK

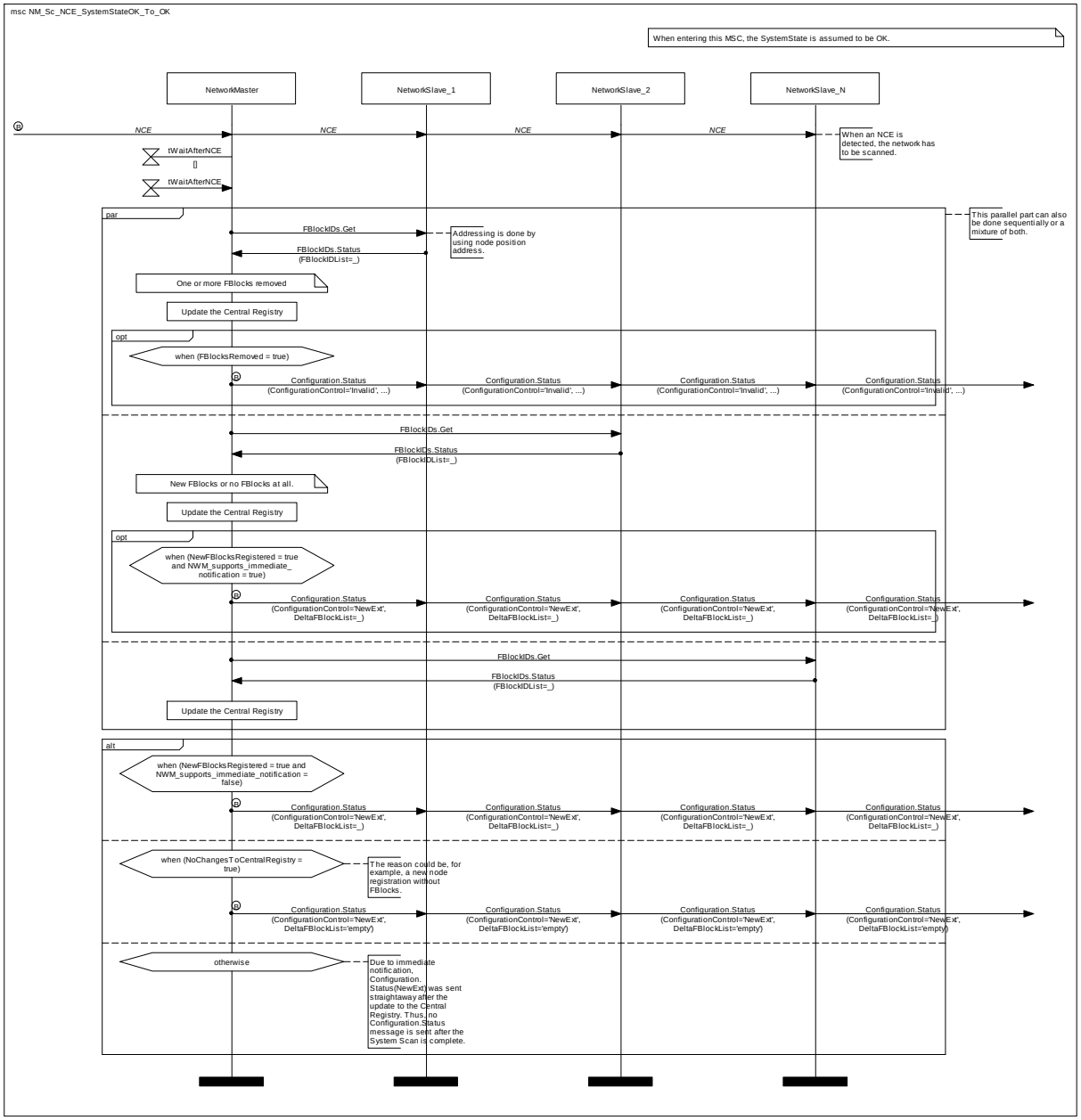
Scenario MSC:	NM_Sc_NCE_SystemStateNotOK_To_OK
Description:	When an NCE occurs, the NetworkMaster has to scan the network (after $t_{WaitAfterNCE}$ has expired). In this scenario, all nodes respond correctly.
Prior Condition:	NetInterface Normal Operation
Initiator:	Any node switching its bypass
Events	NCE
Timers/Timing constraints	$t_{WaitAfterNCE}$
Remarks:	–



MSC 19: NM_Sc_NCE_SystemStateNotOK_To_OK

3.2.3.12 NCE in System State OK Resulting in System State OK

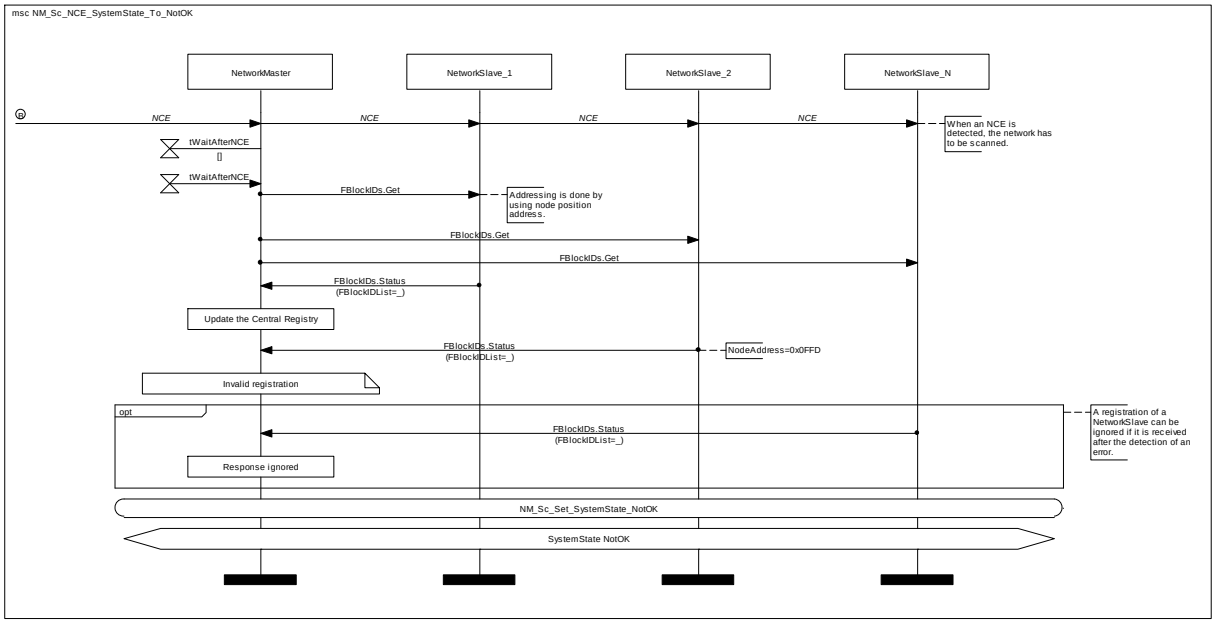
Scenario MSC:	NM_Sc_NCE_SystemStateOK_To_OK
Description:	When an NCE occurs, the NetworkMaster has to scan the network (after $t_{WaitAfterNCE}$ has expired). In this scenario, all nodes respond correctly.
Prior Condition:	NetInterface Normal Operation
Initiator:	Any node switching its bypass
Events	NCE
Timers/Timing constraints	$t_{WaitAfterNCE}$
Remarks:	–



MSC 20: NM_Sc_NCE_SystemStateOK_To_OK

3.2.3.13 NCE Resulting in System State NotOK

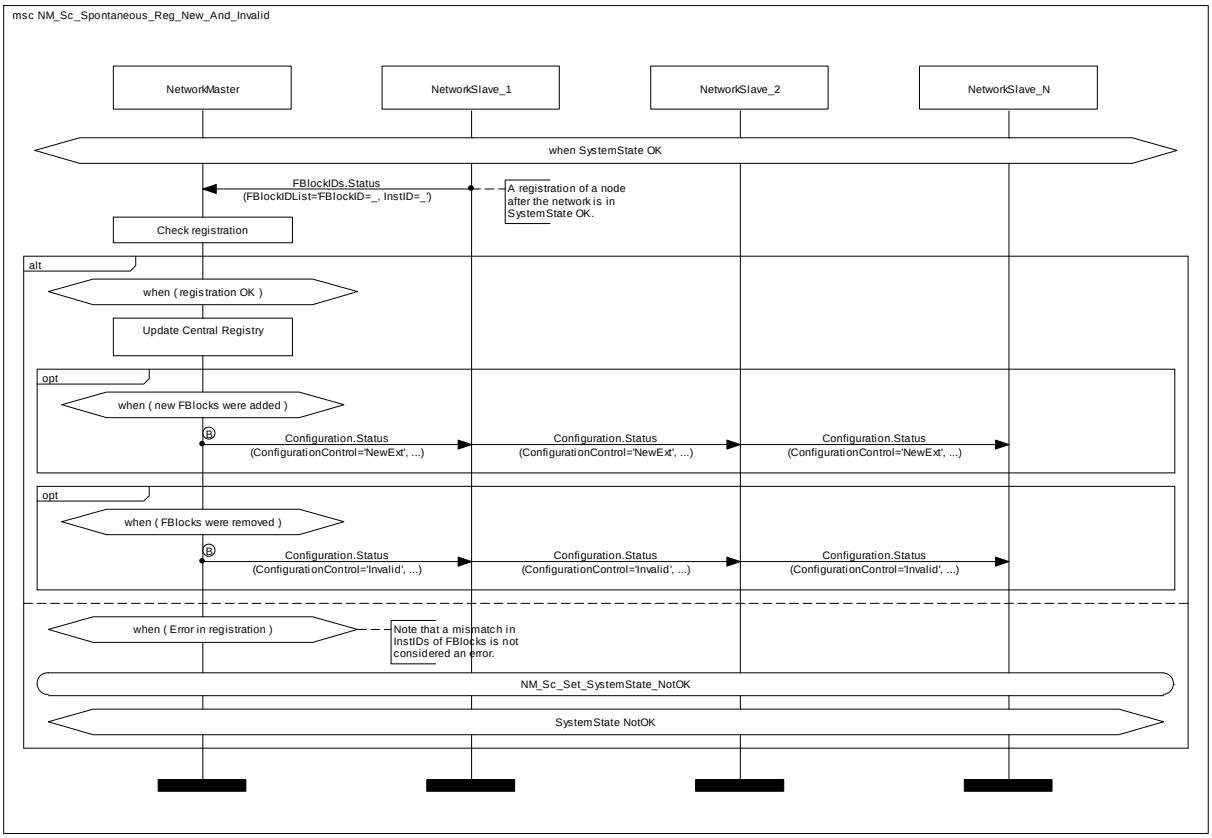
Scenario:	NM_Sc_NCE_SystemState_To_NotOK
Description:	When an NCE occurs, the NetworkMaster has to scan the network (after $t_{WaitAfterNCE}$ has expired). In this scenario, NetworkSlave_2 makes an invalid registration.
Prior Condition:	–
Initiator:	Any node switching its bypass
Events	NCE
Timers/Timing constraints	$t_{WaitAfterNCE}$
Remarks:	– This scenario is valid for all System States. – This scenario is only valid for the mechanism of parallel scanning of the system. It does not cover sequential scanning.



MSC 21: NM_Sc_NCE_SystemState_To_NotOK

3.2.3.14 Spontaneous Registration of a Node

Scenario MSC:	NM_Sc_Spontaneous_Reg_New_And_Invalid
Description:	NetworkSlave_1 makes a spontaneous registration. If new FBlocks are added, a Configuration.Status(NewExt) will be broadcast. If FBlocks are removed, a Configuration.Status(Invalid) will be broadcast. If the registration is invalid, the system will change state to NotOK.
Prior Condition:	System State OK
Initiator:	NetworkSlave_1
Events	–
Timers/Timing constraints	–
Remarks:	–



MSC 22: NM_Sc_Spontaneous_Reg_New_And_Invalid

3.3 Variables used in NetworkSlave MSCs

The use of variables aims at simplifying the MSCs. Table 3-2 shows a list of the variables used in the general Connection Management MSCs.

Variable	Range	Explanation
nodeState	Undefined, SystemCommunicationInit	This variable determines which NetworkInterface state the node is in.
CR_Contains_NWSlave_X	True, False	This variable determines whether a particular node is contained in the Central Registry.

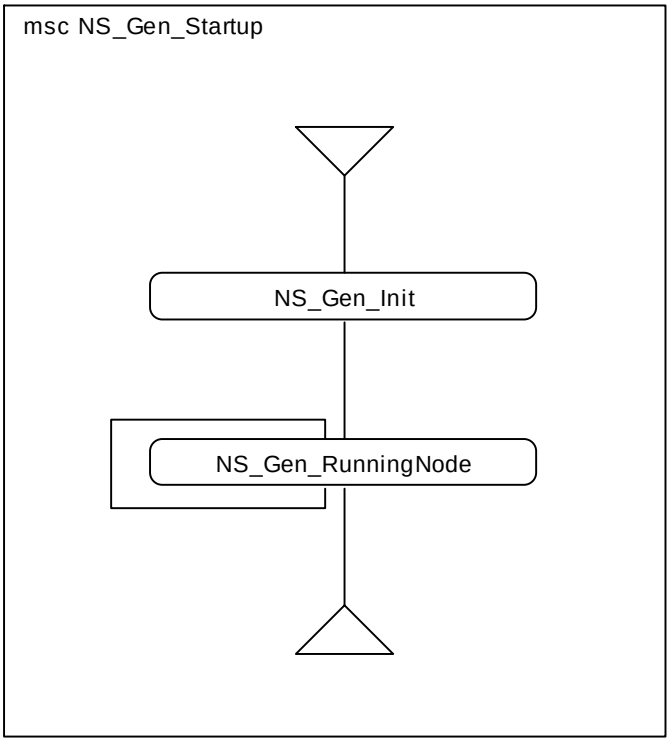
Table 3-2: Variables used in the NetworkSlave MSCs

3.4 NetworkSlave General MSCs

The general MSCs in this section describe the complete startup sequence, from initialization to normal operation, from the perspective of a NetworkSlave. The high-level MSC shows how the general MSCs are combined to describe the complete flow from startup to normal (running) operation in a NetworkSlave.

3.4.1 High-level NetworkSlave MSC

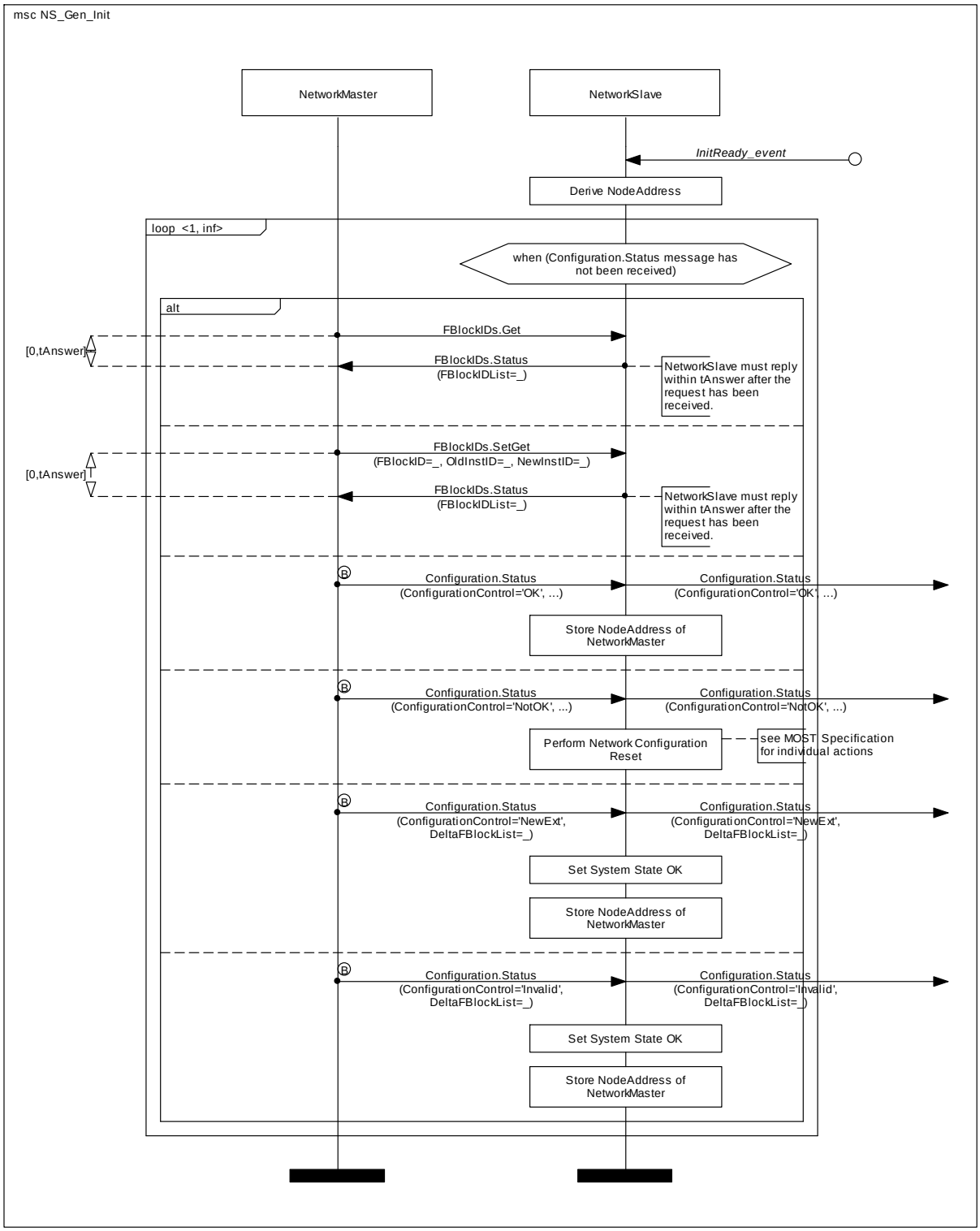
General MSC:	NS_Gen_Startup
Description:	High-level MSC of NetworkSlave startup process. The startup sequence for MOST Nodes is described in this high-level MSC.
Prior Condition:	–
Initiator:	–
Events:	Init Ready
Timers/Timing constraints:	–
Remarks:	–



MSC 23: NS_Gen_Startup

3.4.2 Initializing the NetworkSlave

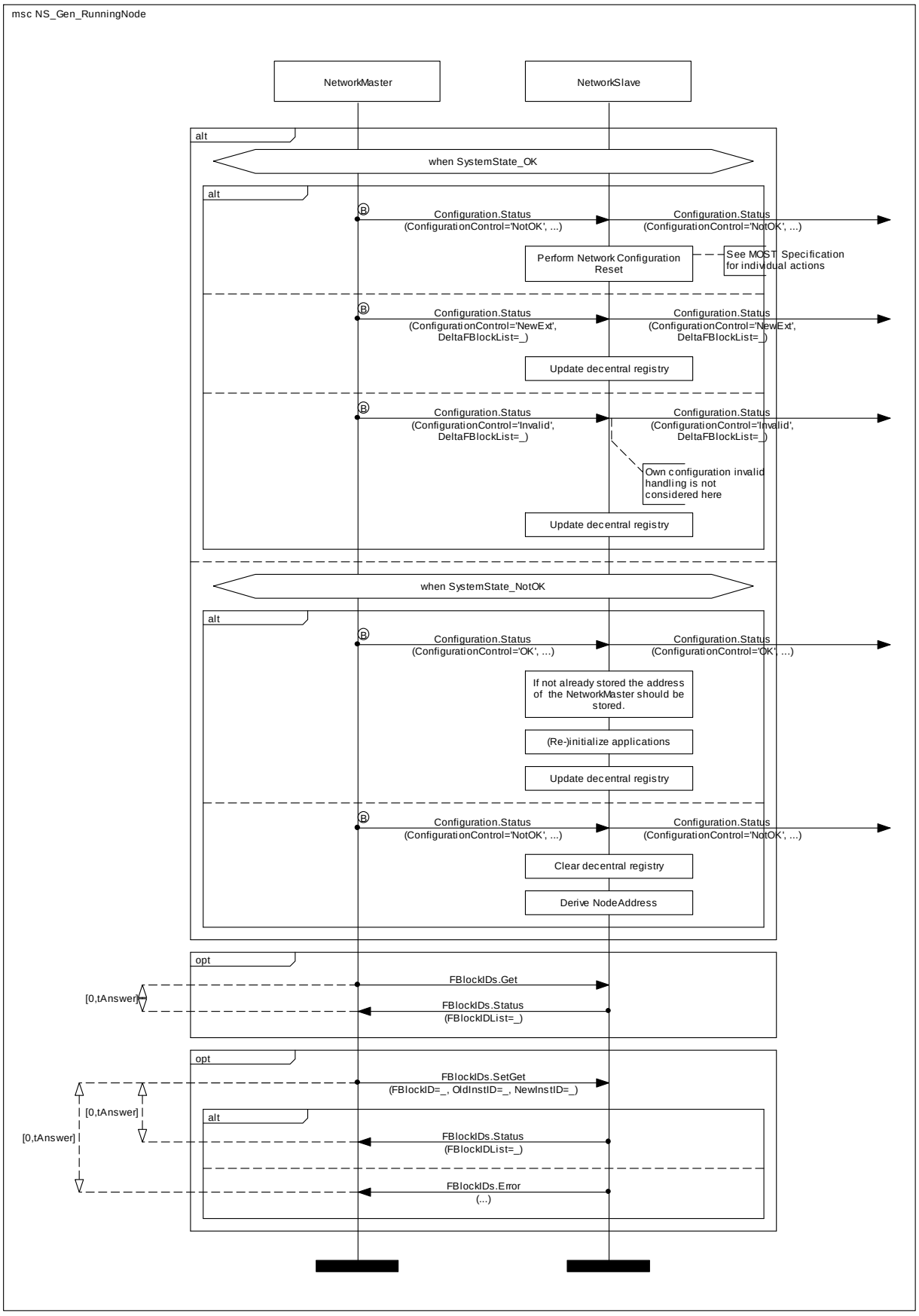
General MSC:	NS_Gen_Init
Description:	Describes the initialization and startup sequence of a NetworkSlave after Init Ready.
Prior Condition:	NetInterface Init
Initiator:	–
Events:	Init Ready
Timers/Timing constraints:	t_{Answer}
Remarks:	–



MSC 24: NS_Gen_Init

3.4.3 Node Running Operation

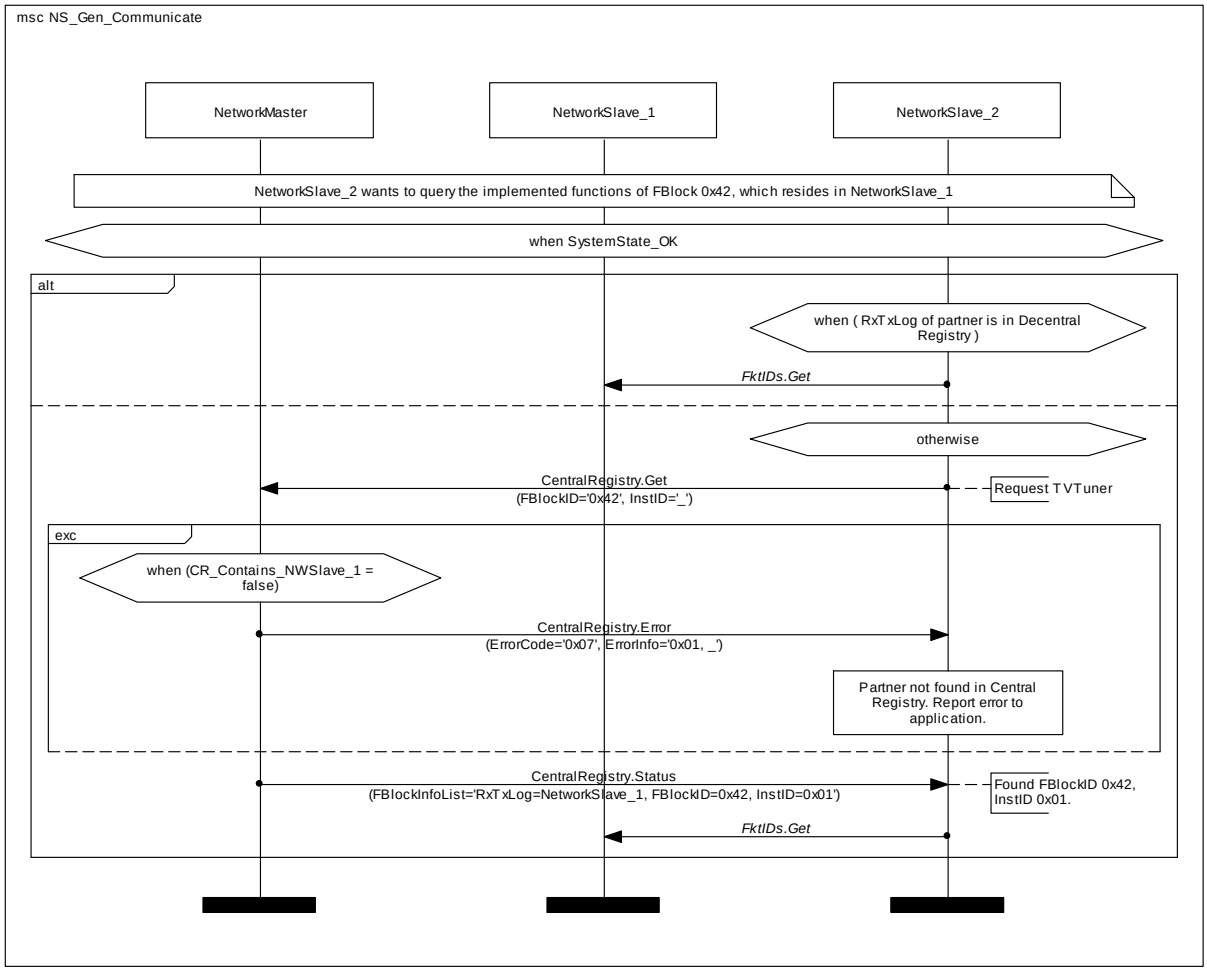
General MSC:	NS_Gen_RunningNode
Description:	Describes the running (normal) operation of a NetworkSlave after the startup sequence has been completed.
Prior Condition:	NetInterface Normal Operation
Initiator:	NetworkMaster
Events:	–
Timers/Timing constraints:	t_{Answer}
Remarks:	–



MSC 25: NS_Gen_RunningNode

3.4.4 Communicate

General MSC:	NS_Gen_Communicate
Description:	Describes how a NetworkSlave uses its Decentral Registry or the Central Registry to find the logical address of its communication partner.
Prior Condition:	System State OK
Initiator:	NetworkSlave_2
Events:	–
Timers/Timing constraints:	–
Remarks:	–



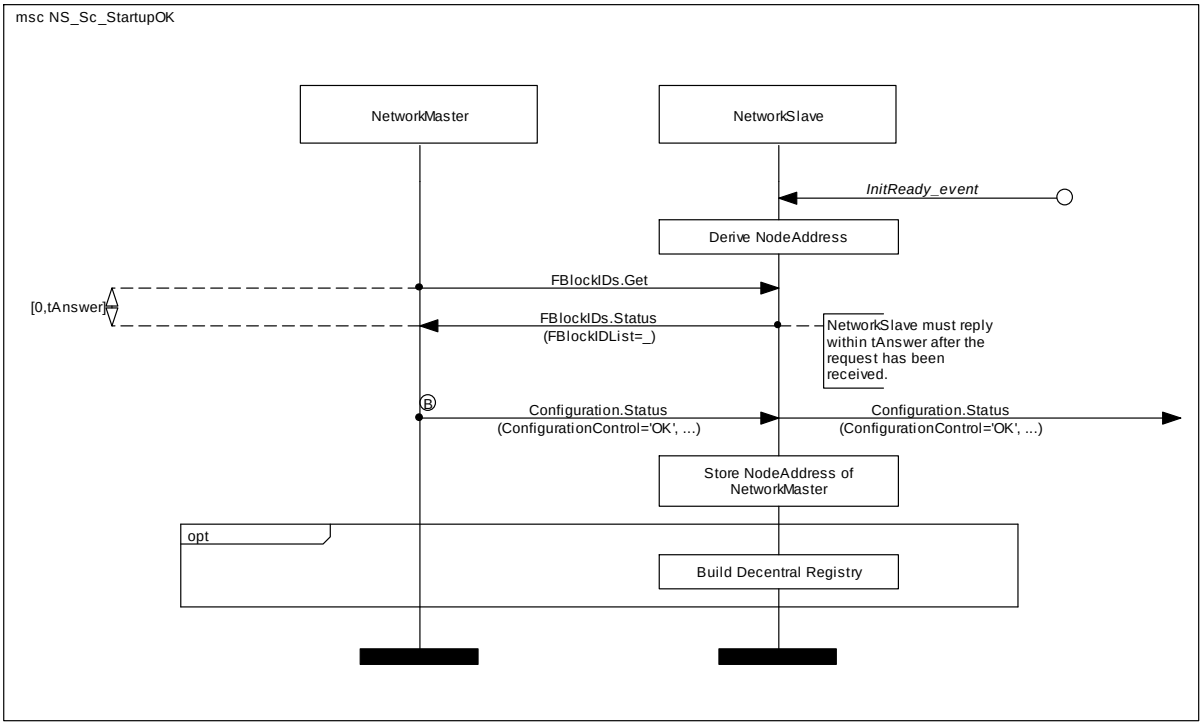
MSC 26: NS_Gen_Communicate

3.5 NetworkSlave Scenario MSCs

3.5.1 Startup scenarios

3.5.1.1 Startup - OK

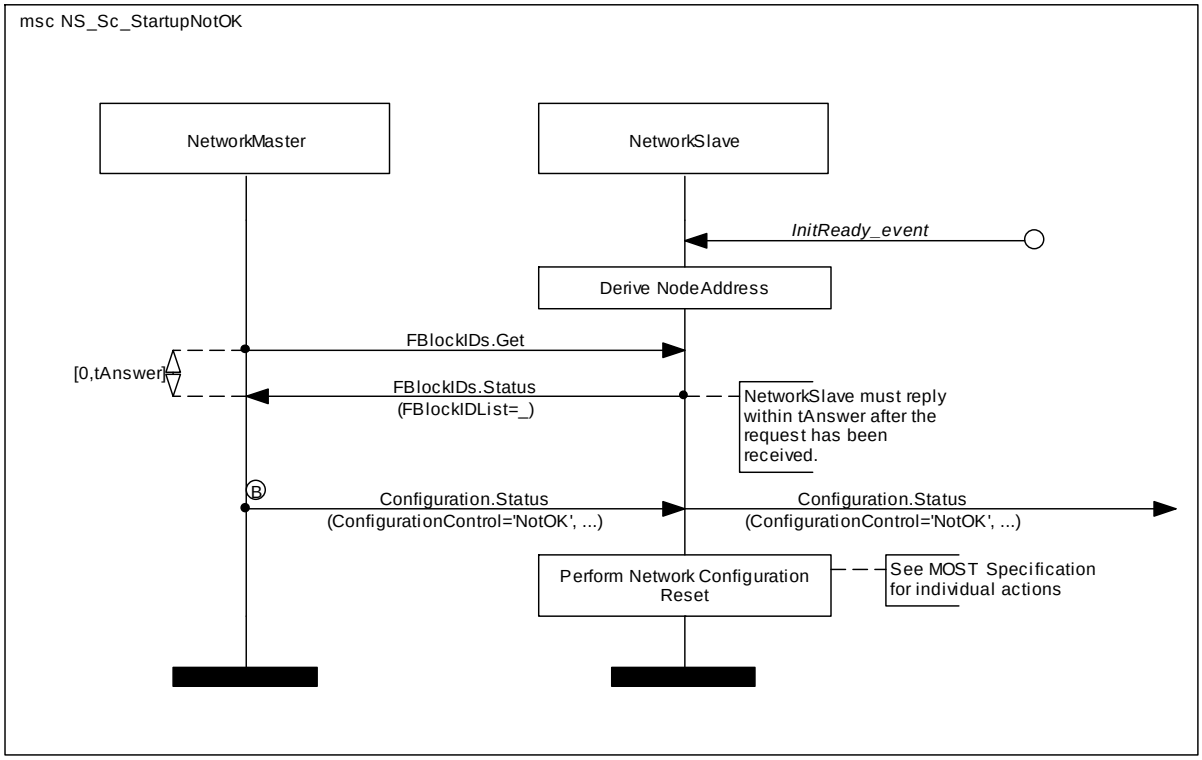
Scenario MSC	NS_Sc_StartupOK
Description:	Describes the startup sequence of a NetworkSlave when Configuration.Status(OK) is received during startup.
Prior Condition:	NetInterface Init
Initiator:	–
Events:	Init Ready
Timers/Timing constraints:	t_{Answer}
Remarks:	–



MSC 27: NS_Sc_StartupOK

3.5.1.2 Startup - NotOK

Scenario MSC	NS_Sc_StartupNotOK
Description:	Describes the startup sequence of a NetworkSlave when Configuration.Status(NotOK) is received during startup.
Prior Condition:	NetInterface Init
Initiator:	–
Events:	Init Ready
Timers/Timing constraints:	t_{Answer}
Remarks:	–

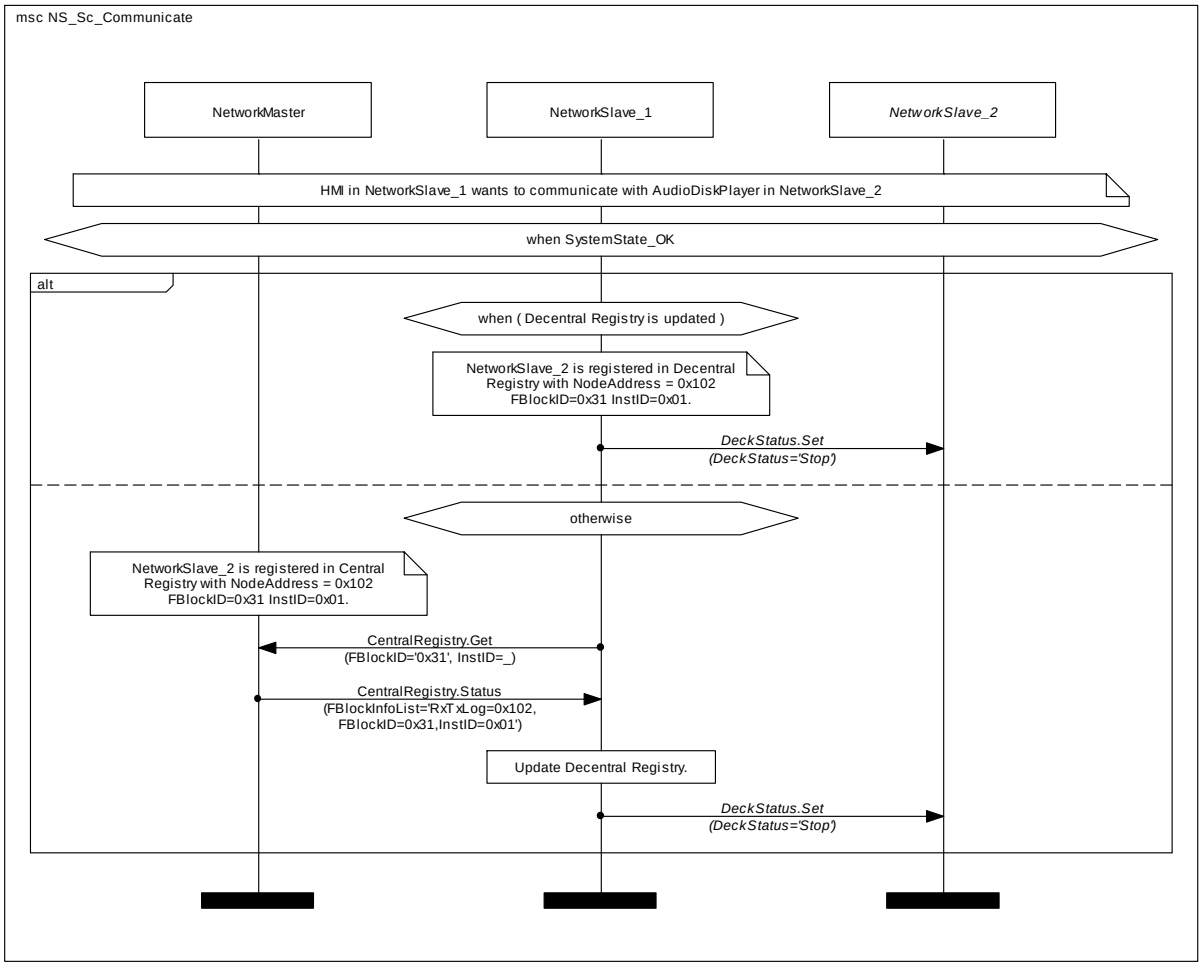


MSC 28: NS_Sc_StartupNotOK

3.5.2 Communication Scenarios

3.5.2.1 Communicate with partner

Scenario MSC	NS_Sc_Communicate
Description:	Describes how a NetworkSlave uses its Decentral Registry or the Central Registry to find the logical address of its communication partner.
Prior Condition:	System State OK
Initiator:	NetworkSlave_1
Events:	–
Timers/Timing constraints:	–
Remarks:	As an example, HMI and AudioDiskPlayer are used as communication partners.



MSC 29: NS_Sc_Communicate

4 Connection Management

4.1 Variables used in Connection Management MSCs

The general MSCs use variables to simplify the MSCs, as well as reducing the total number of MSCs. Table 4-1 shows a list of the variables used in the general Connection Management MSCs.

Variable	Range	Explanation
Error	True, False	Indicates if something fails during connection management.
Progress	None, SourceInfo_Received, Source_Connected, Sink_Connected, SourceActivity_On, Functions_Received	This variable is used to keep track of how far the procedure has progressed.
DisconnectResult	Success, Failure	The result of a disconnect operation in a sink.
AllocateResult	Success, Failure	The result of an allocate operation in a source.
DeallocateResult	Success, Failure	The result of a de-allocate operation in a source.
BoundaryLimit	NotExceeded, Exceeded	This variable is set to Exceeded when a controller issues a MoveBoundary request where the requested boundary is in conflict with active connections.
AnnounceBC	unsupported, supported	This variable determines whether the Connection Management supports a method to announce the start of the boundary shifting process.

Table 4-1: Variables used in the general Connection Management MSCs

4.2 Normal Behavior

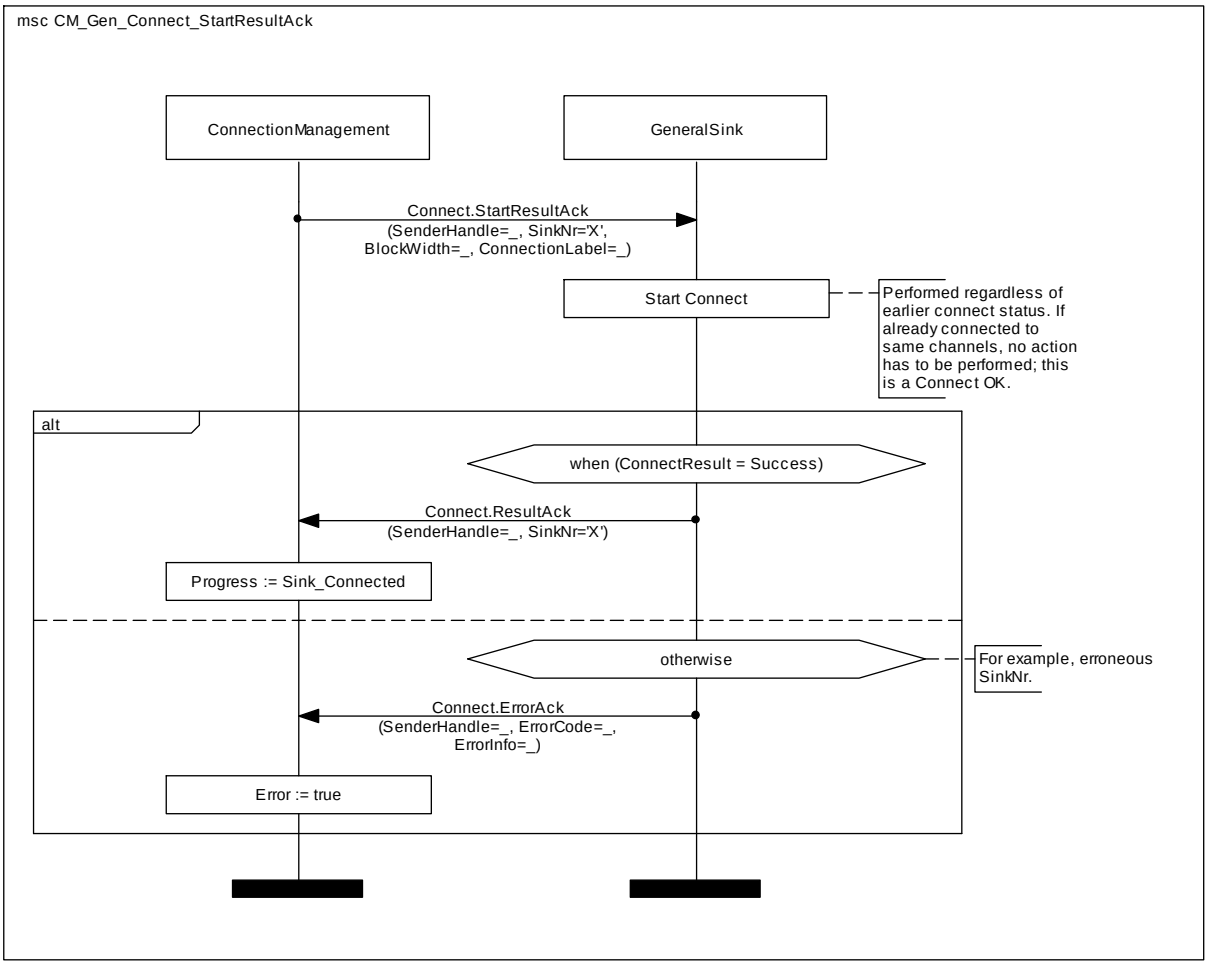
In this section, MSCs are used to describe Connection Management in normal behavior.

When building a connection, it is important whether the Connection Management possesses required information about the nodes.
When this is the case, the Connection Management does not have to inquire how many channels different sources need. The Connection Management may have obtained this information either by previously querying the nodes or by having the information already provided by the system developers.

4.2.1 Connection Management General MSCs

4.2.1.1 Connecting a Sink

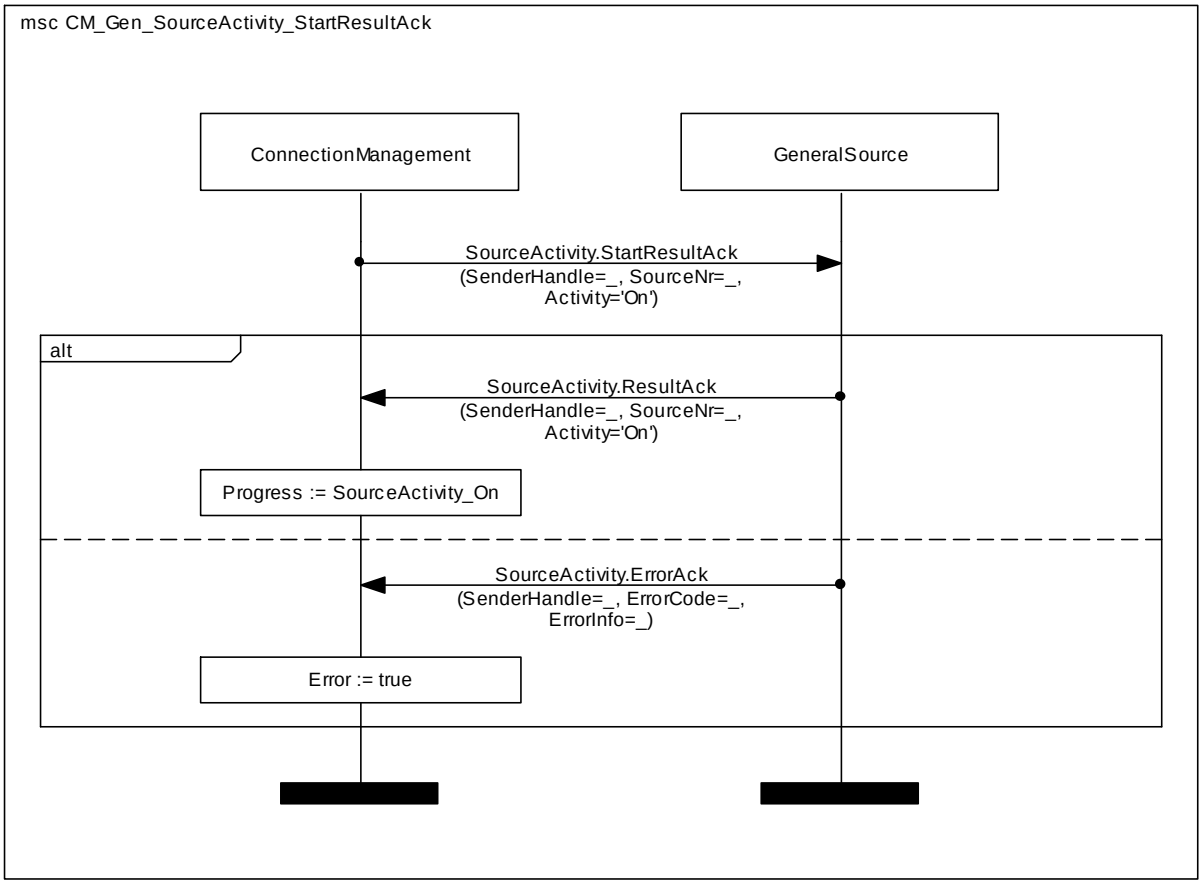
General MSC:	CM_Gen_Connect_StartResultAck
Description:	The Connection Management commands the sink to connect to the specified channels. The result of this MSC is saved for the higher level MSC that uses this one.
Prior Condition:	The channels that the sink is connecting to are in use by a source.
Initiator:	Connection Management
Events	–
Timers / Timing Constraints:	–
Remarks:	–



MSC 30: CM_Gen_Connect_StartResultAck

4.2.1.2 SourceActivity turned on

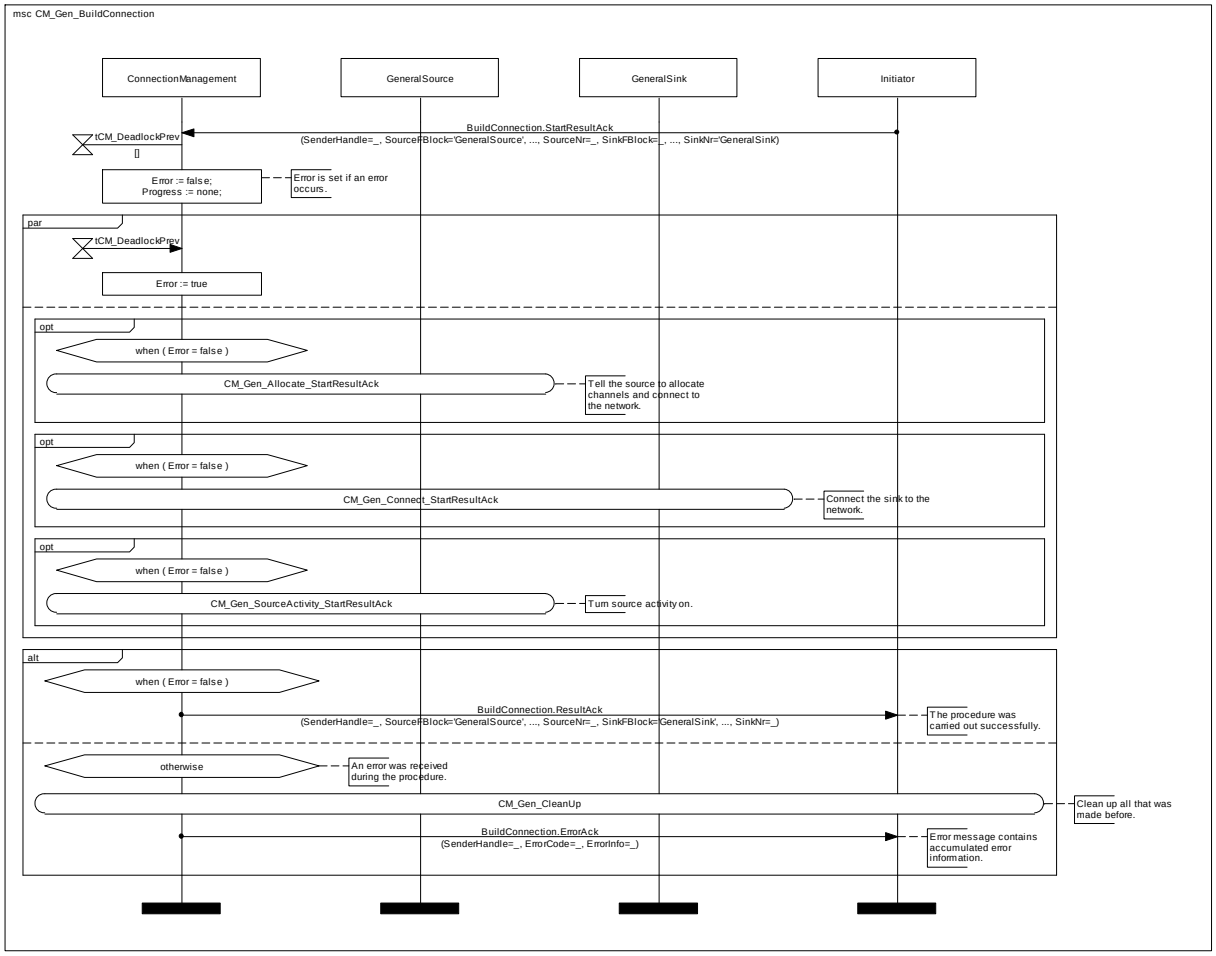
General MSC:	CM_Gen_SourceActivity_StartResultAck
Description:	The Connection Management commands the source to turn the SourceActivity on. The result of this MSC is saved for the higher level MSC that uses this one.
Prior Condition:	A connection to a sink has been established.
Initiator:	Connection Management
Events	–
Timers / Timing Constraints:	–
Remarks:	The SourceActivity function is optional.



MSC 31: CM_Gen_SourceActivity_StartResultAck

4.2.1.3 BuildConnection

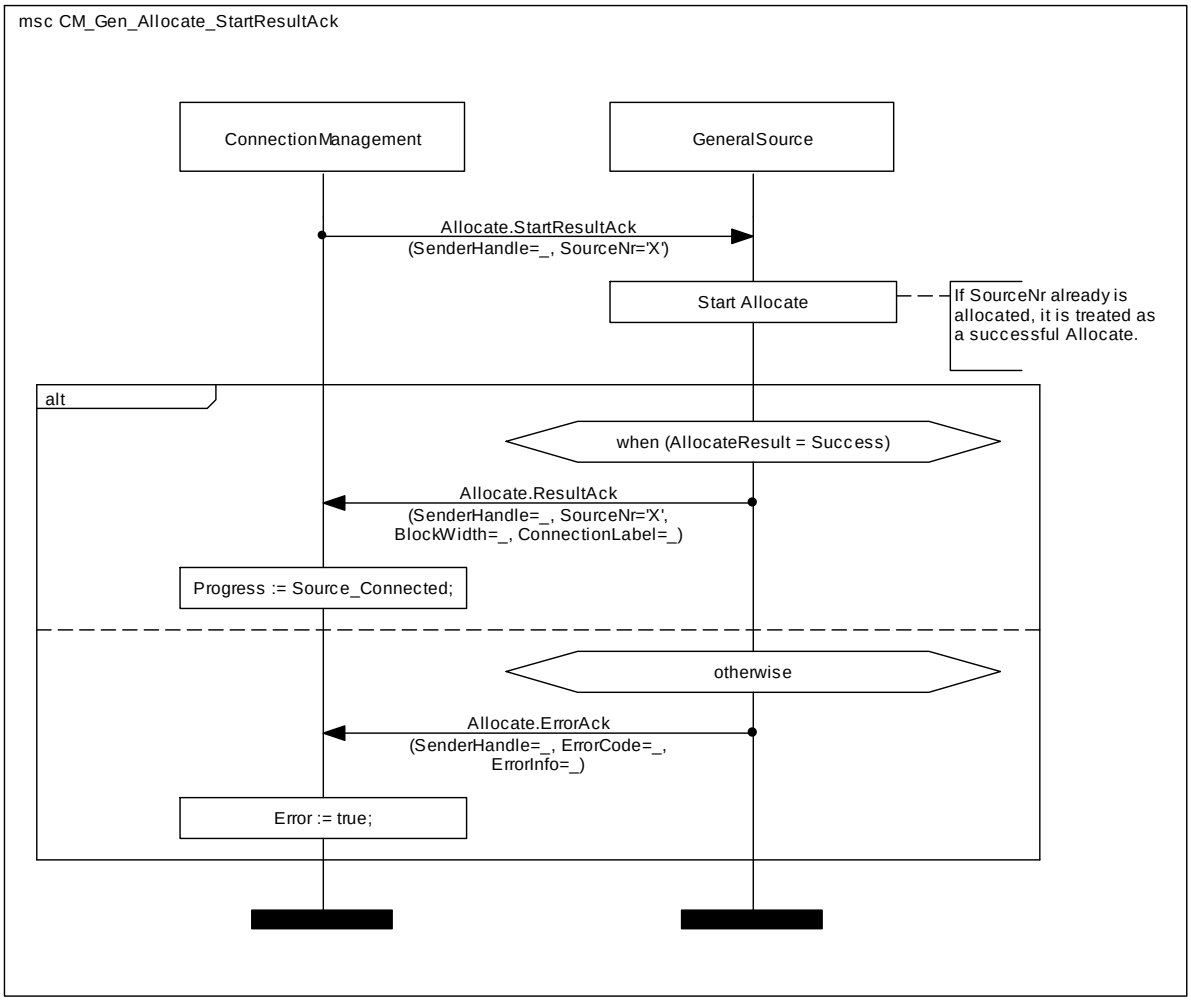
General MSC:	CM_Gen_BuildConnection
Description:	A connection is built between GeneralSource and GeneralSink. The flow is as follows: The source is connected to the network and the sink is also connected. When the connection is established, SourceActivity may be turned on. A timer is started when the Connection Manager starts this process. If it takes more than the $t_{CM\_DeadlockPrev}$ to do this, the process is aborted. The Connection Management will have to tidy up after itself by removing everything that has been done so far, which is done in the CleanUp MSC.
Prior Condition:	–
Initiator:	Any Controller
Events	–
Timers / Timing Constraints:	$t_{CM\_DeadlockPrev}$
Remarks:	– While running CleanUp, there is no reaction to an incoming AbortAck. The same applies after SourceActivity has been turned on. – The SourceActivity function is optional.



MSC 32: CM_Gen_BuildConnection

4.2.1.4 Allocate

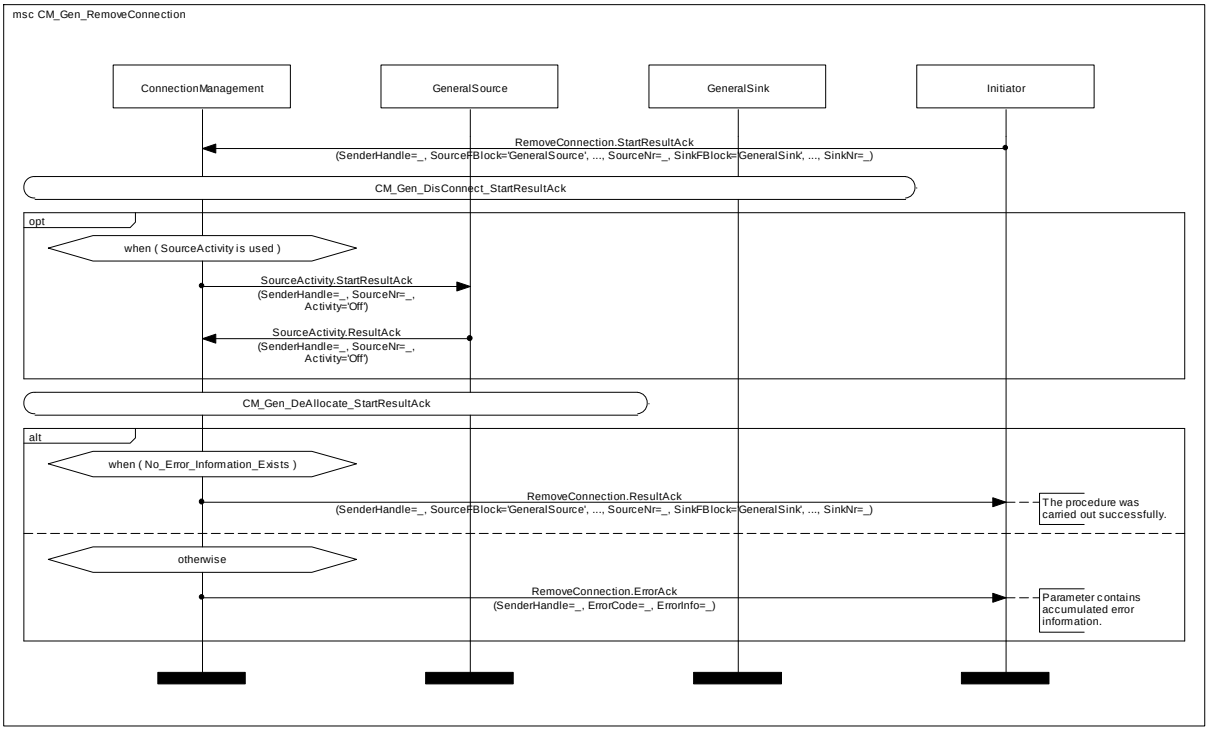
General MSC:	CM_Gen_Allocate_StartResultAck
Description:	The Connection Management tells the source to allocate bandwidth and to connect. The result of this MSC is saved for the higher level MSC that uses this one.
Prior Condition:	–
Initiator:	Connection Management
Events:	–
Timer / Timing Constraints:	–
Remarks:	If the optional function SourceActivity is used, the source must not start routing data before SourceActivity.StartResult(On) is called.



MSC 33: CM_Gen_Allocate_StartResultAck

4.2.1.5 Removing a Synchronous Connection

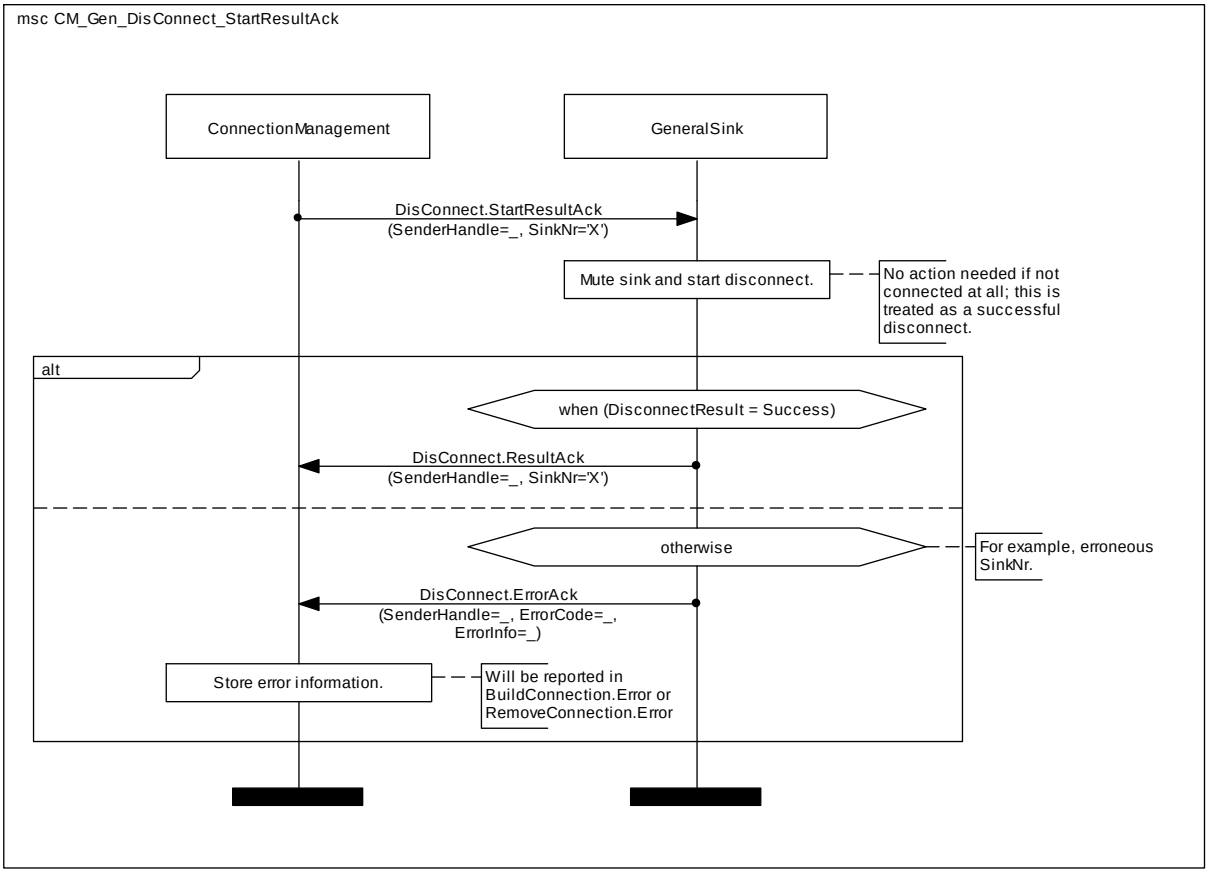
General MSC:	CM_Gen_RemoveConnection
Description:	The Connection Management removes a connection. First, the sink is disconnected (and muted). Then the source is disconnected with DeAllocate. When sending this command, SourceActivity is turned off (if used).
Prior Condition:	A connection between the source and sink exists.
Initiator:	Any Controller
Events	–
Timer / Timing Constraints:	–
Remarks:	– There is no way of aborting this procedure. – The SourceActivity function is optional.



MSC 34: CM_Gen_RemoveConnection

4.2.1.5.1 Disconnecting a Sink

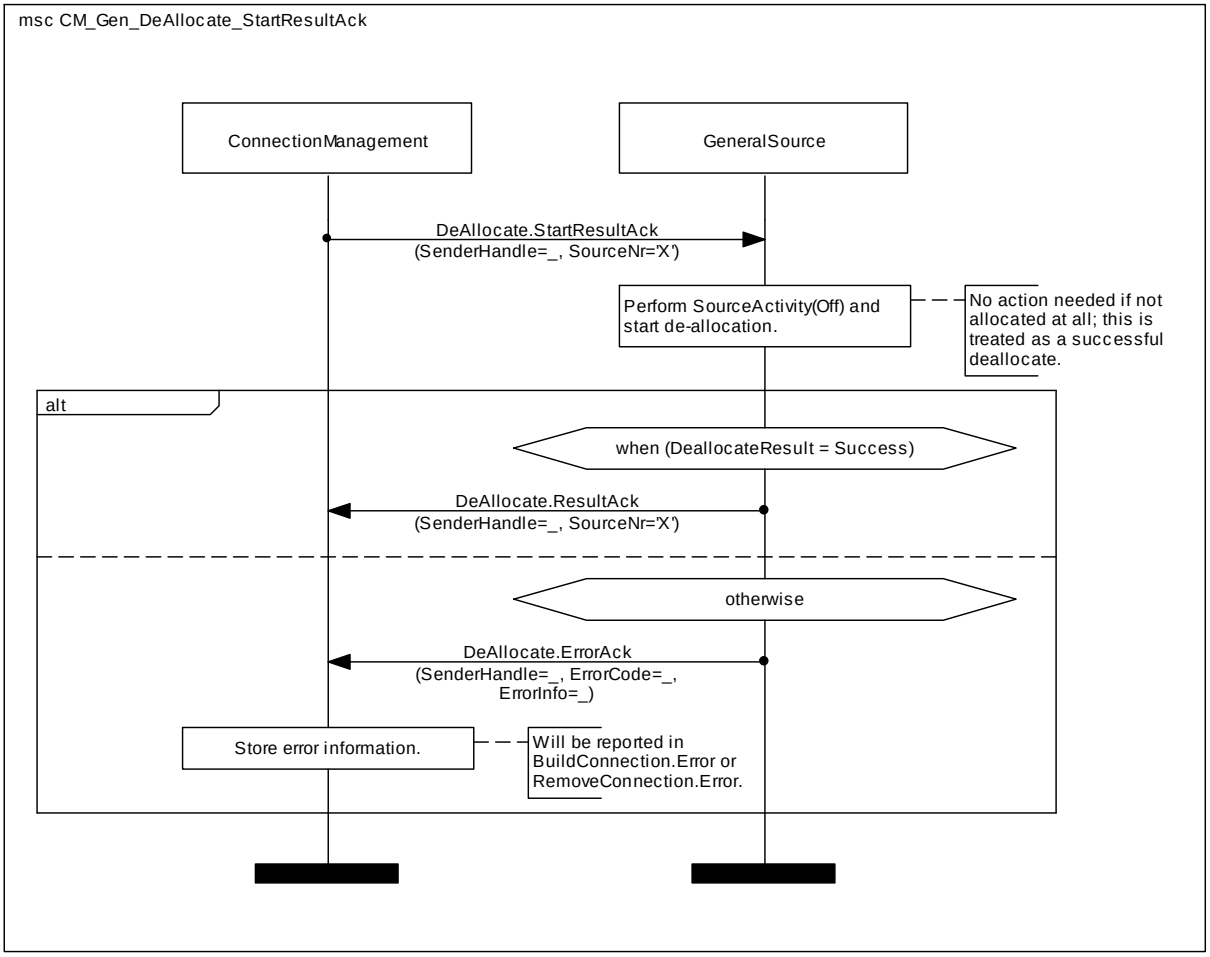
General MSC:	CM_Gen_DisConnect_StartResultAck
Description:	Connection Management tells the sink to disconnect the specified sink. Error information is saved for the MSC that uses this one.
Prior Condition:	The sink is connected to the network.
Initiator:	Connection Management
Events	–
Timer / Timing Constraints:	–
Remarks:	–



MSC 35: CM_Gen_DisConnect_StartResultAck

4.2.1.5.2 Deallocation Procedure

General MSC:	CM_Gen_DeAllocate_StartResultAck
Description:	Connection Management tells the source to deallocate the specified channels and to disconnect. Error information is saved for the MSC that uses this one.
Prior Condition:	The source is connected to the network and uses the specified channels.
Initiator:	Connection Management
Events	–
Timer / Timing Constraints:	–
Remarks:	– The source stops routing and removes its channels.



MSC 36: CM_Gen_DeAllocate_StartResultAck

4.3 DiscreteFrame Isochronous Connection Handling

4.3.1 Data Source and Phase Source in One Device

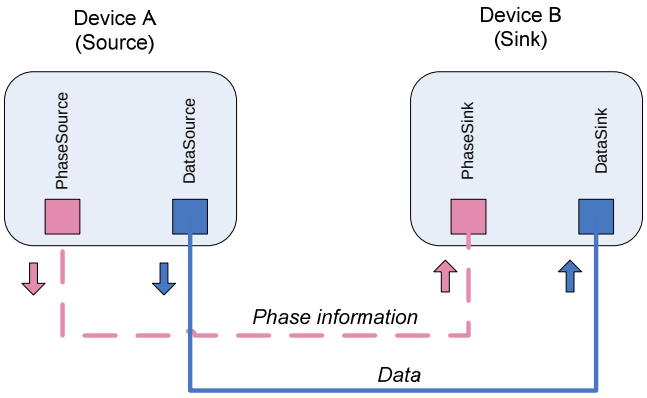
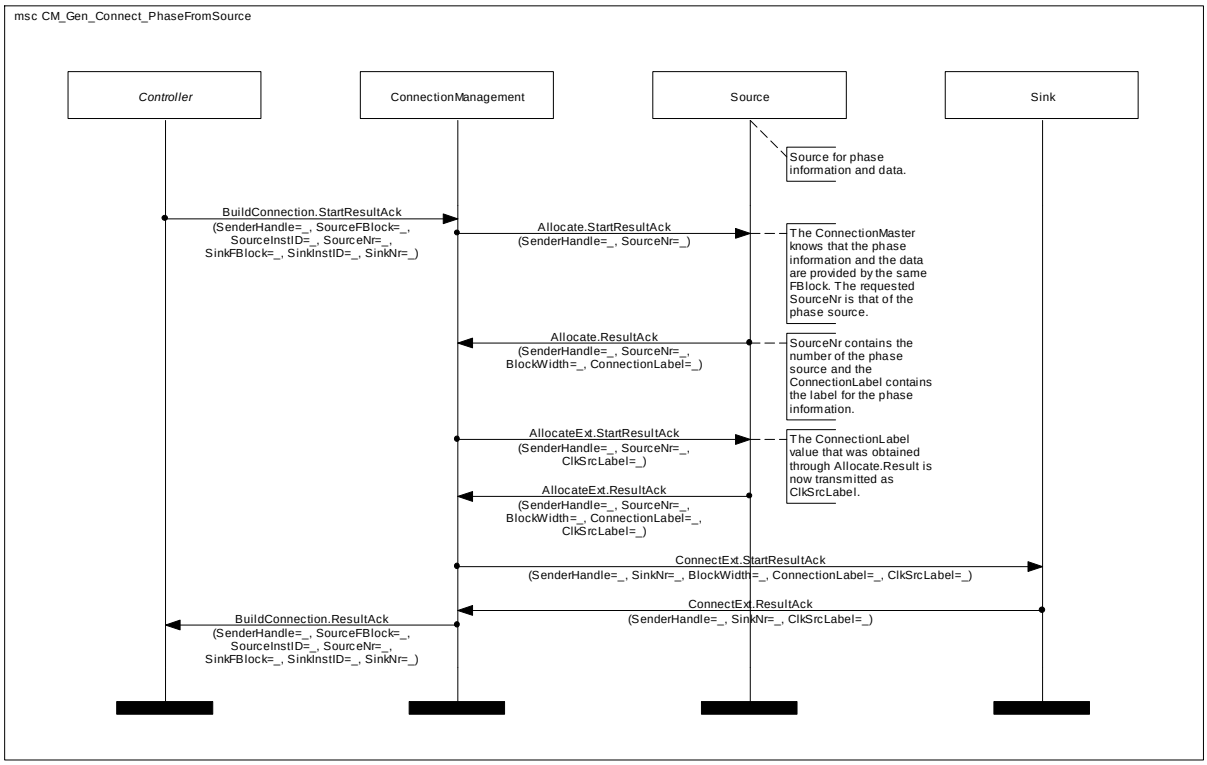


Figure 4-1: DiscreteFrame Isochronous – data source and phase source in one device

General MSC:	CM_Gen_Connect_PhaseFromSource
Description:	A Controller requests a connection between a source that offers DiscreteFrame Isochronous data and a sink that can process that kind of data. The source offers both the phase information and the payload.
Prior Condition:	System State OK
Initiator:	Controller
Events	–
Timer / Timing Constraints:	–
Remarks:	–



MSC 37: CM_Gen_Connect_PhaseFromSource

4.3.2 Phase Information from a Third Device

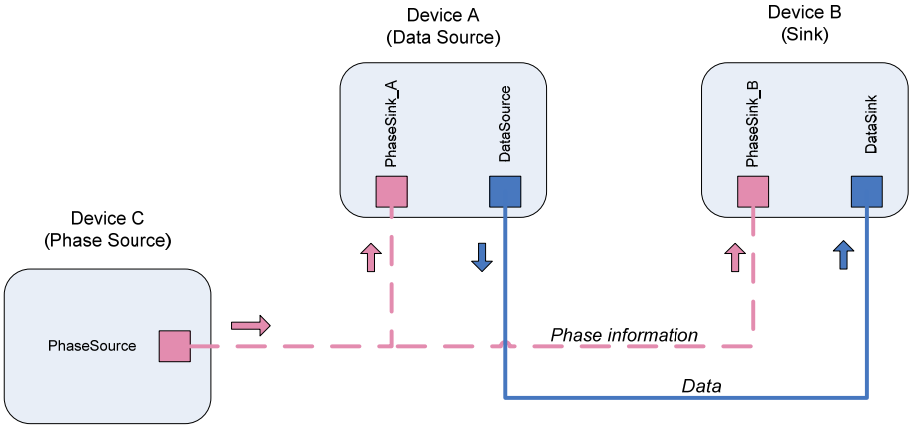
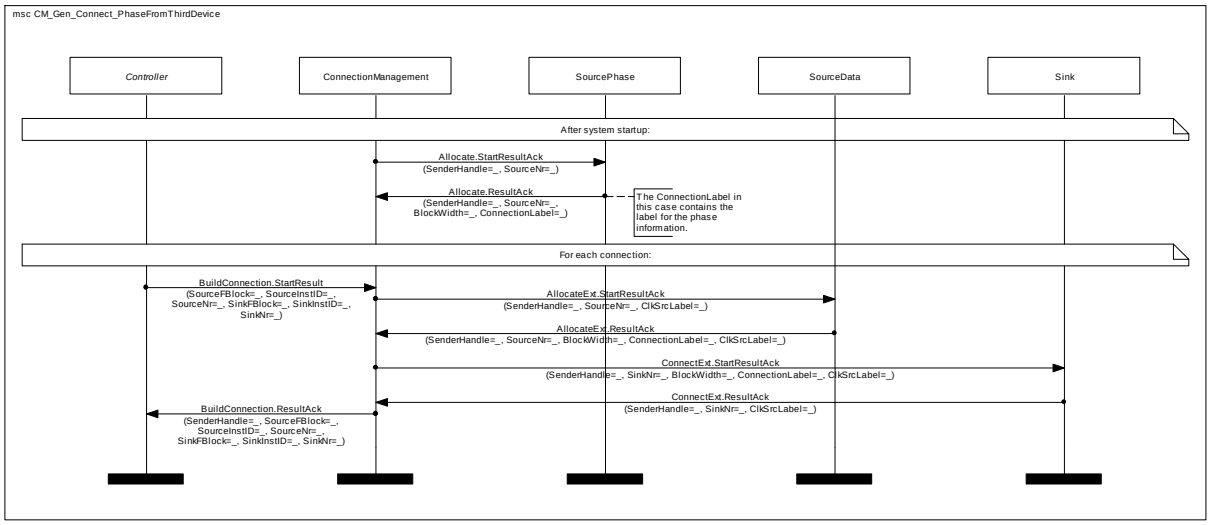


Figure 4-2: DiscreteFrame Isochronous – phase information form a third device

General MSC:	CM_Gen_Connect_PhaseFromThirdDevice
Description:	A Controller requests a connection between a source that offers DiscreteFrame Isochronous data and a sink that can process that kind of data. The phase information and the payload are provided by different sources. The ConnectionMaster sets up the phase source after system startup, independent of any requests.
Prior Condition:	System State OK
Initiator:	Controller
Events	–
Timer / Timing Constraints:	–
Remarks:	–



MSC 38: CM_Gen_Connect_PhaseFromThirdDevice

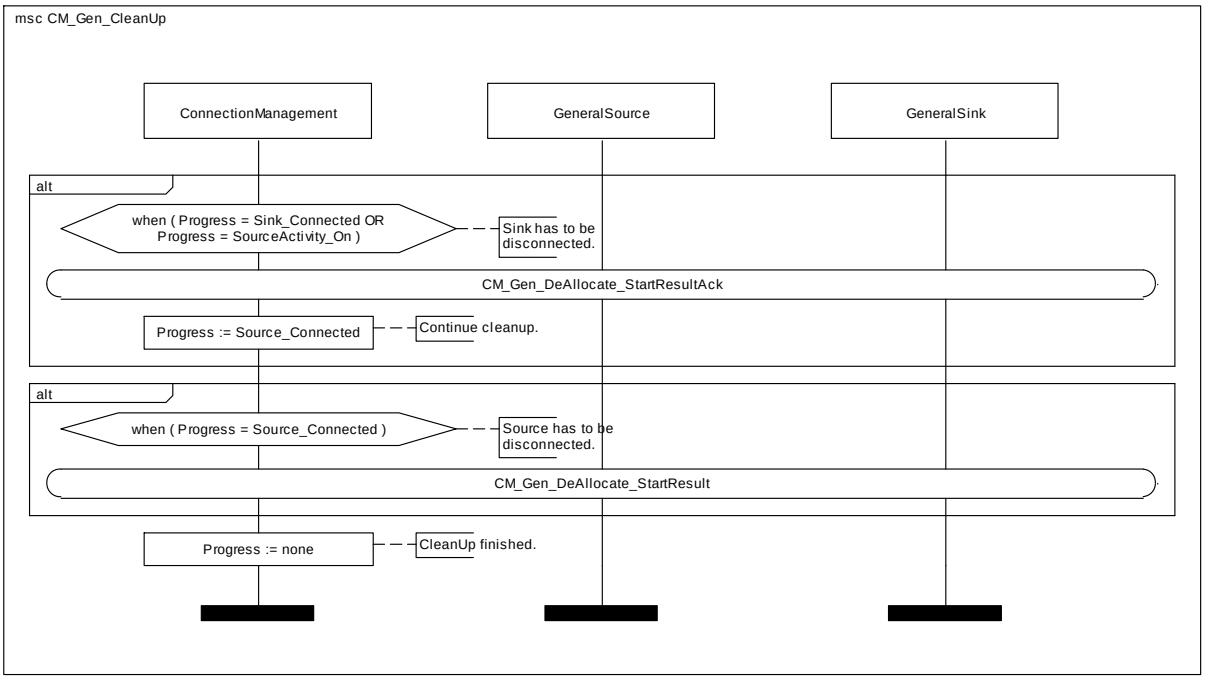
4.4 Error Handling

This section describes the behavior of the Connection Management when dealing with error cases.

4.4.1 Error Handling General MSCs

4.4.1.1 CleanUp

General MSC:	CM_Gen_CleanUp
Description:	Connection Management unmakes everything that has been made in the connection procedure.
Prior Condition:	An error or abort has interrupted the BuildConnection procedure.
Initiator:	This MSC is initiated after an abort or error within BuildConnection.
Events	–
Timer / Timing Constraints:	–
Remarks:	There is no way of aborting the CleanUp process.

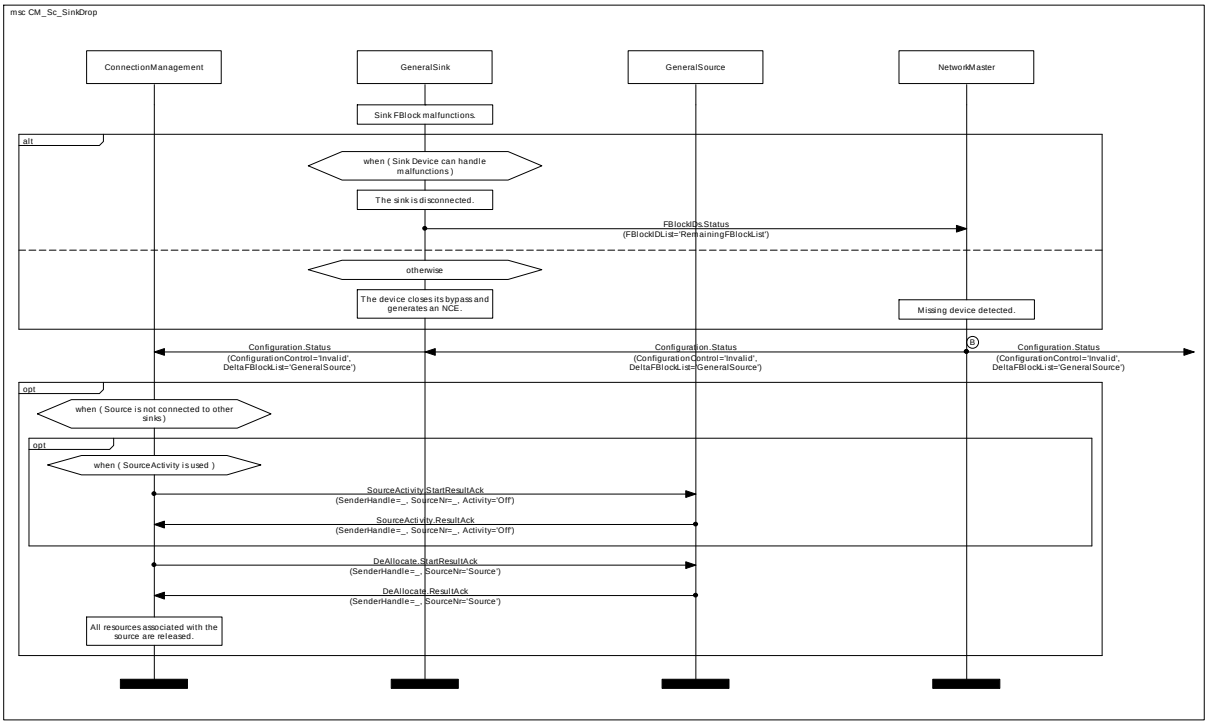


MSC 39: CM_Gen_CleanUp

4.4.2 Error Handling Scenario MSCs

4.4.2.1 Sink drop

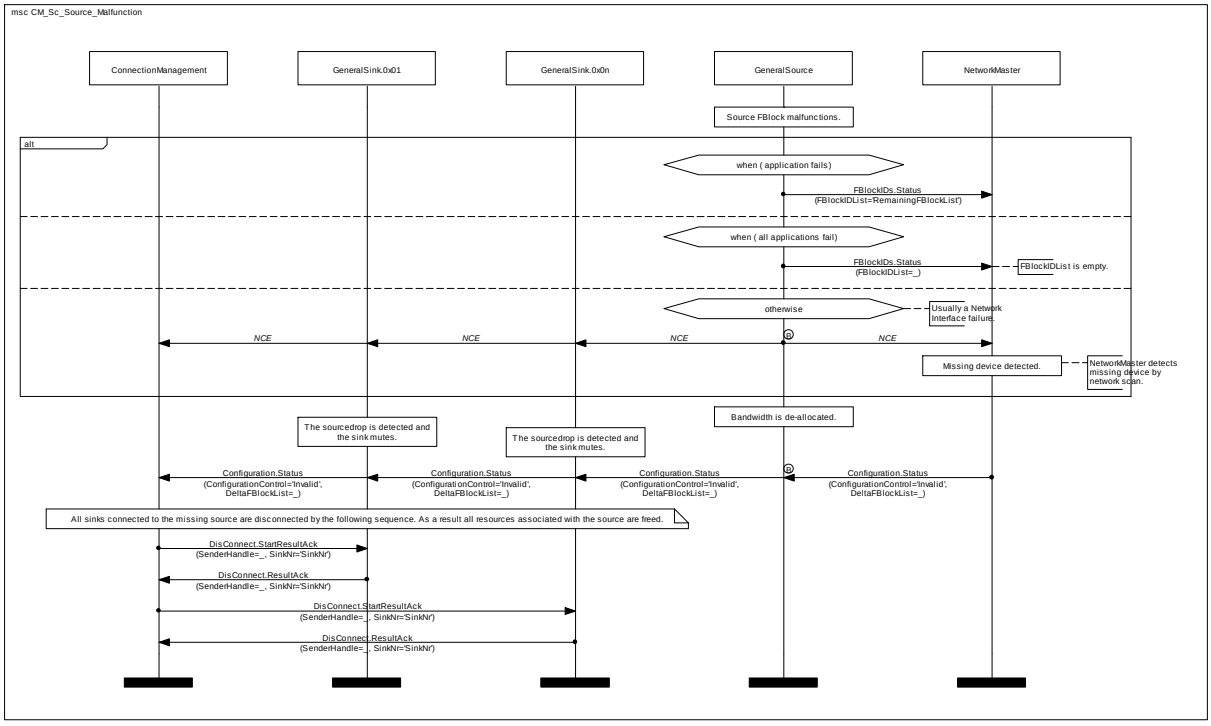
General MSC:	CM_Sc_SinkDrop
Description:	General Sink drops out due to an unlikely internal error.
Prior Condition:	–
Initiator:	Any failing sink
Events	–
Timer / Timing Constraints:	–
Remarks:	The SourceActivity function is optional.



MSC 40: CM_Sc_SinkDrop

4.4.2.2 Source Malfunction

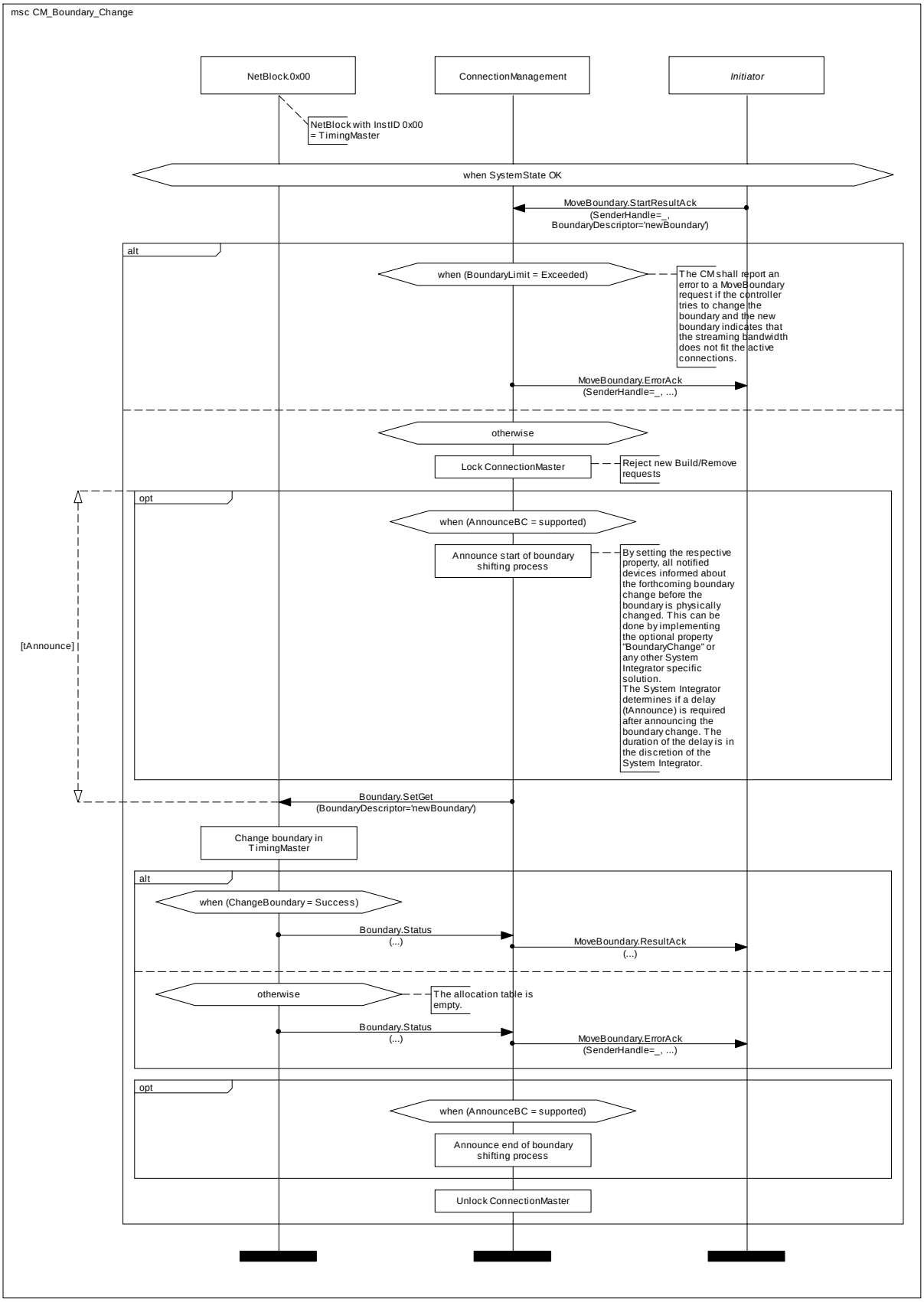
General MSC:	CM_Sc_Source_Malfunction
Description:	General Source drops out due to an unlikely internal error.
Prior Condition:	–
Initiator:	Any failing source
Events	–
Timer / Timing Constraints:	–
Remarks:	–



MSC 41: CM_Sc_Source_Malfunction

4.5 Boundary Change

Use Case:	CM_Boundary_Change
Description:	The initiator requests a Boundary change from the ConnectionMaster, which forwards the request to the TimingMaster.
Prior Condition:	Configuration Status OK
Initiator:	– Initiator
Events:	–
Remarks:	–



MSC 42: CM_Boundary_Change

5 Power management

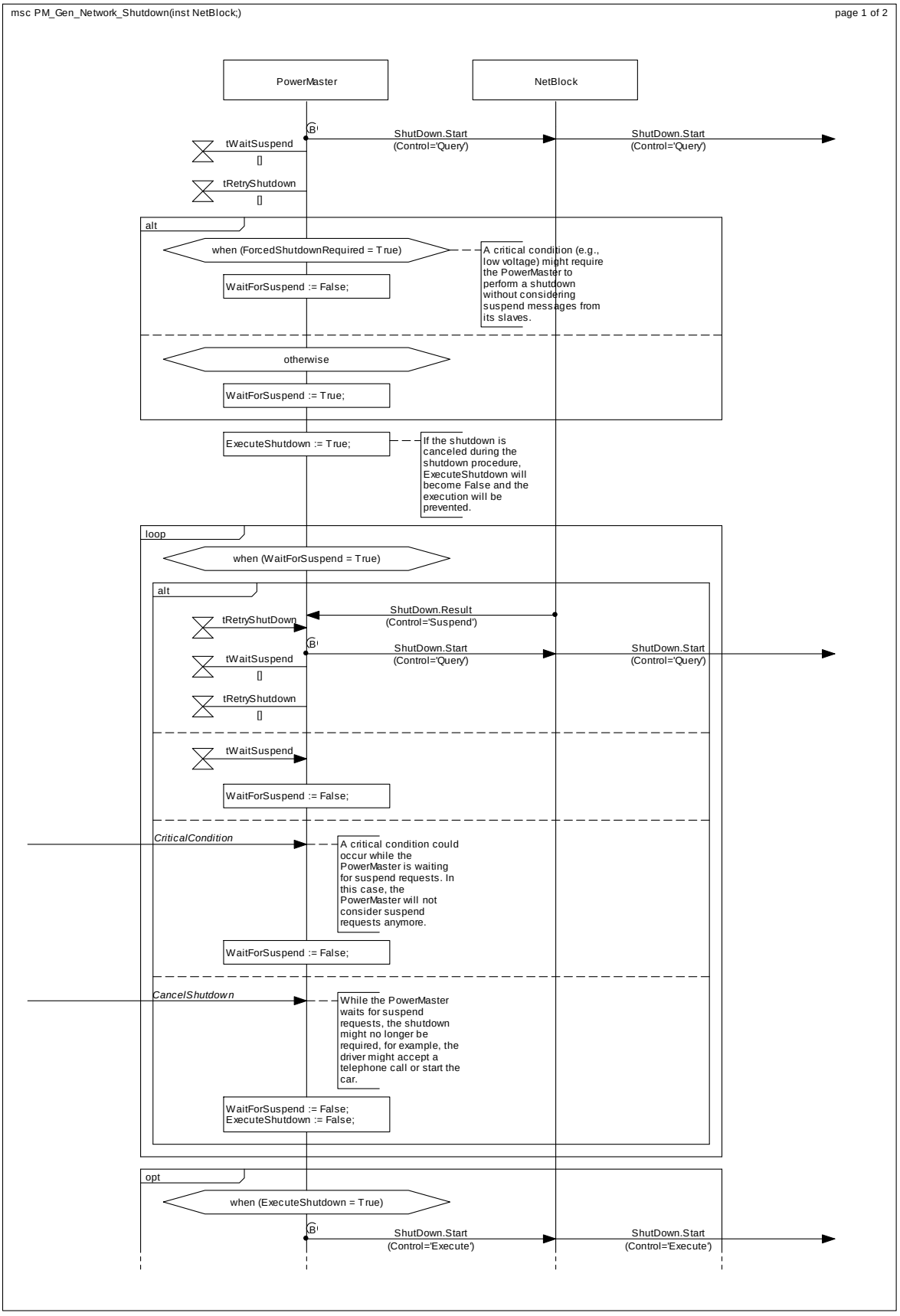
5.1 Introduction

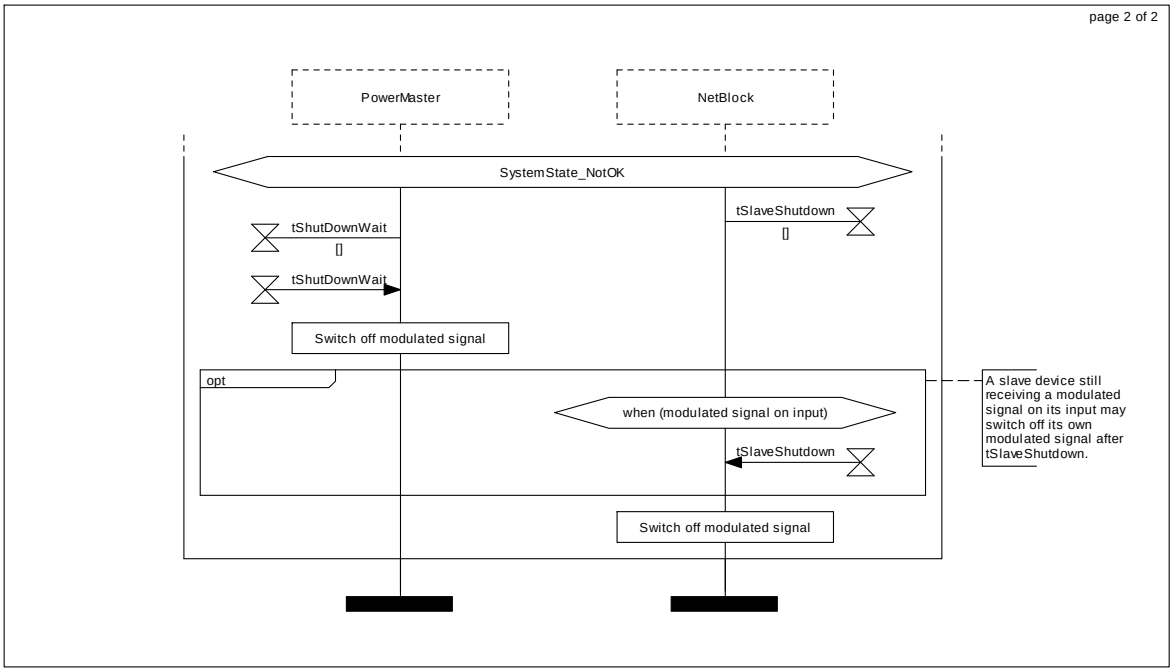
Power management means that the administrative function, which is above the Network Service, wakes and shuts down the MOST network or specific devices. The power management is handled mainly by the PowerMaster, which uses NetBlock functions for this purpose.

Switching on the network is described in the MOST Specification.

5.2 Network shutdown

General MSC:	PM_Gen_Network_Shutdown
Description:	This process handles a shutdown of the network.
Prior Condition:	NetInterface Normal Operation
Initiator:	– PowerMaster
Events:	–
Timers / Timing Constraints:	– $t_{WaitSuspend}$ – $t_{RetryShutDown}$ – $t_{ShutDownWait}$ – $t_{SlaveShutDown}$
Remarks:	–

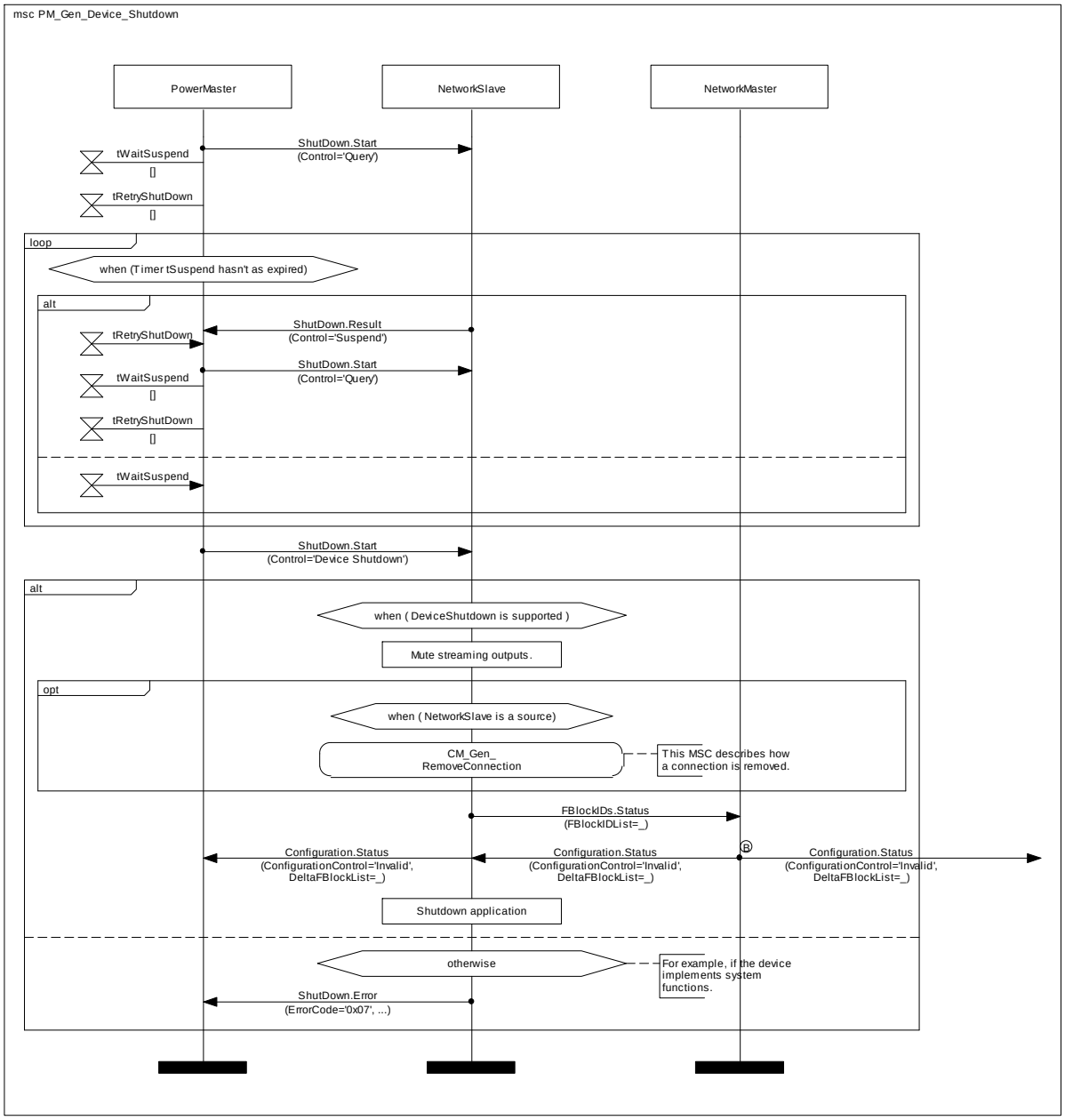




MSC 43: PM_Gen_Network_Shutdown

5.3 Device shutdown

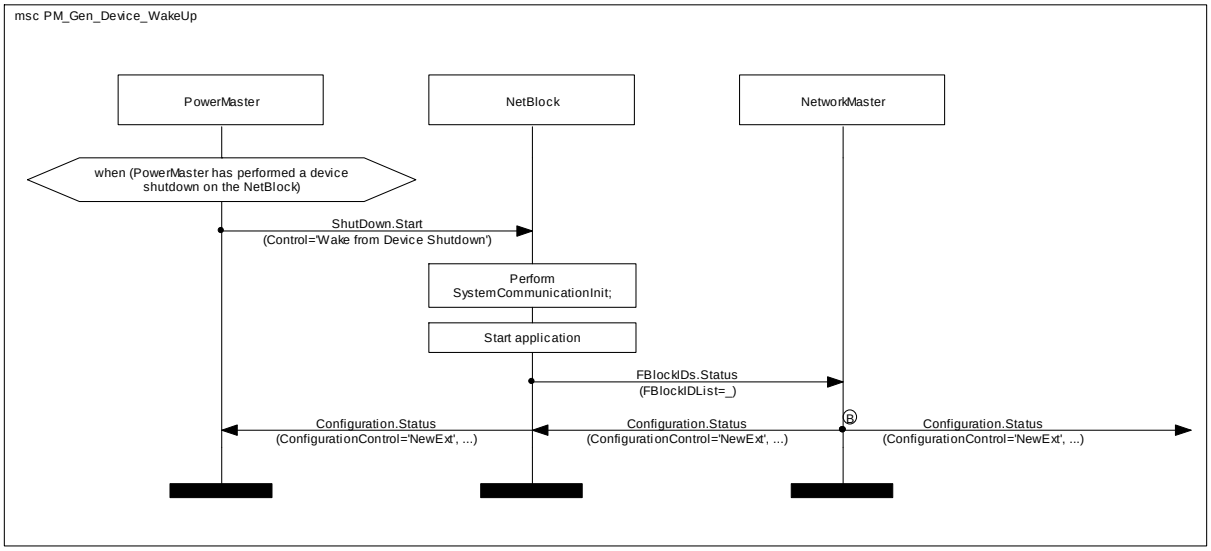
General MSC:	PM_Gen_Device_Shutdown
Description:	This process handles a shutdown of a device that is initiated by the PowerMaster.
Prior Condition:	NetInterface Normal Operation
Initiator:	PowerMaster
Events:	–
Timers / Timing Constraints:	– $t_{WaitSuspend}$ – $t_{RetryShutDown}$
Remarks:	–



MSC 44: PM_Gen_Device_Shutdown

5.4 Device wake up

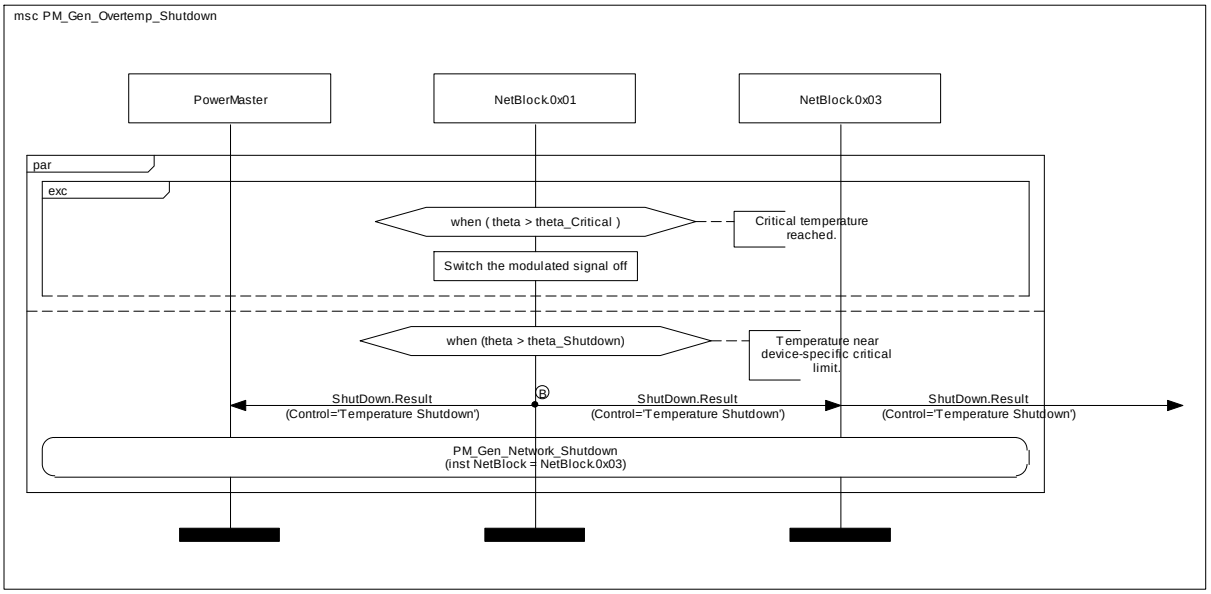
General MSC:	PM_Gen_Device_WakeUp
Description:	This process handles a wake up of a device that is initiated by the PowerMaster.
Prior Condition:	NetInterface Normal Operation
Initiator:	PowerMaster
Events:	–
Timers / Timing Constraints:	–
Remarks:	–



MSC 45: PM_Gen_Device_WakeUp

5.5 Network shutdown due to over-temperature

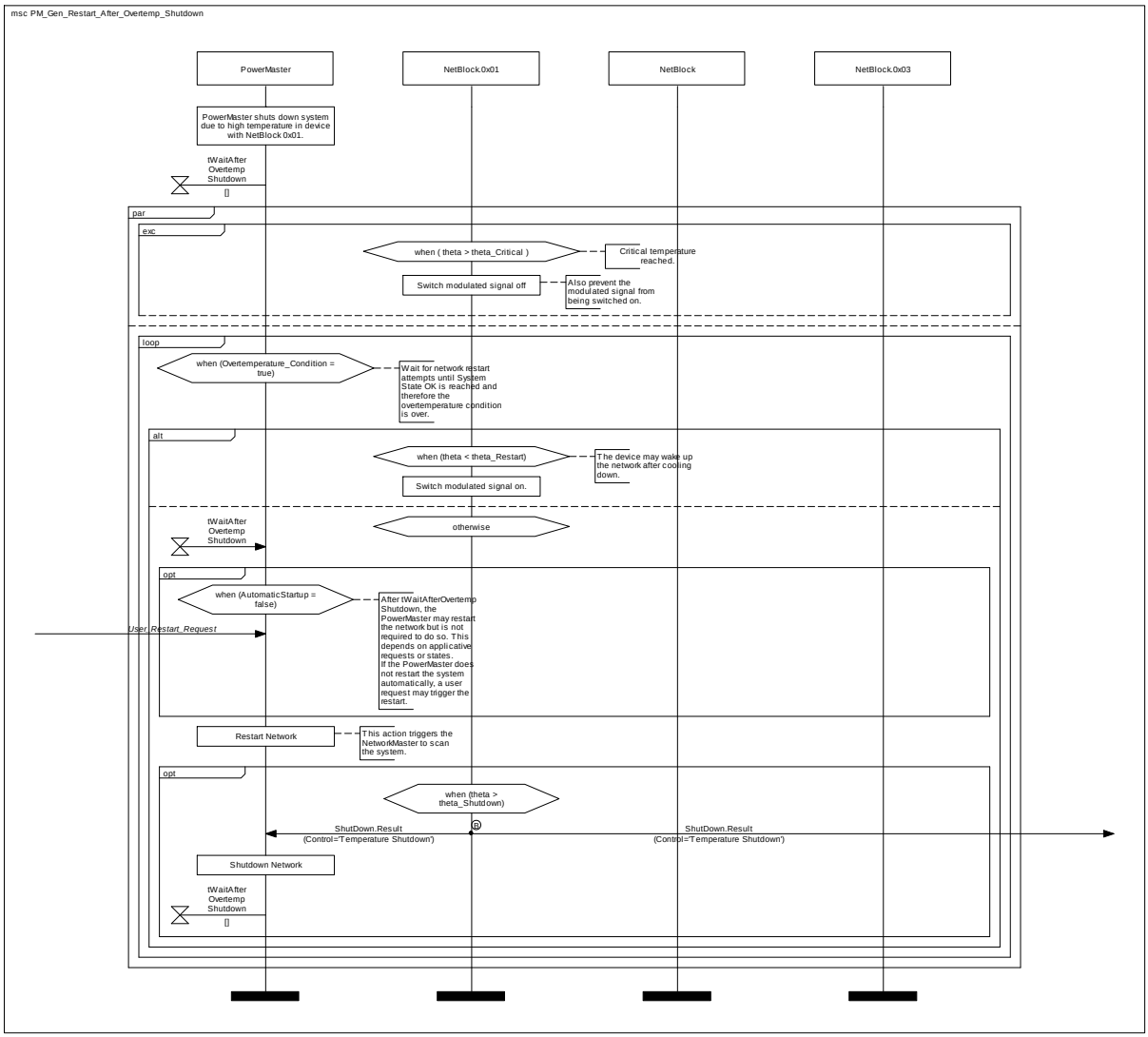
General MSC:	PM_Gen_Overtemp_Shutdown
Description:	This process handles a shutdown of the network due to over temperature. This scenario applies if the PowerMaster implements the optional feature to perform a normal shutdown after receiving ShutDown.Result(0x03).
Prior Condition:	NetInterface Normal Operation
Initiator:	NetBlock in device that is overheated.
Events:	–
Timers / Timing Constraints:	–
Remarks:	–



MSC 46: PM_Gen_Overtemp_Shutdown

5.6 Network restart after over-temperature shutdown

General MSC:	PM_Gen_Restart_After_Overtemp_Shutdown
Description:	This process handles a network restart after a shutdown of the network due to over-temperature.
Prior Condition:	–
Initiator:	PowerMaster
Events:	–
Timers / Timing Constraints:	$t_{WaitAfterOvertempShutdown}$
Remarks:	–



MSC 47: PM_Gen_Restart_After_Overtemp_Shutdown

6 Timers

The definition of timers can be found in the corresponding MOST Specification.

7 Naming Conventions

The names of the MSCs categorize them into different sections. Every name has a prefix that differentiates it from MSCs dealing with other topics. Two parts make up the prefix. The first part consists of an abbreviation or a characteristic name of the topic that the MSC focuses on. The following names exist:

Topic Prefix	Contents of the MSC
CM	Connection Management MSC
NM	NetworkMaster MSC
NS	NetworkSlave MSC
PM	Power Management MSC

Table 7-1: Topic Prefixes for MSCs

The second part of the prefix is used to differentiate between General MSCs and Scenario MSCs. _Gen_ stands for general and _Sc_ for Scenario.

8 Appendix A: Index of Figures

9 Appendix B: Index of Tables

10 Appendix C: Index of MSCs

Notes: