Functional Tests - Security Manager

Generated by Doxygen 1.8.6

Tue Sep 29 2015 15:26:54

Contents

Chapter 1

Test List

Class secmgr_tests::AJNCaStorageTest_BasicTest_Test

Basic tests for the sample implementation of a CAStorage based on AllJoyn.

1. Initialize an AJNCAStorage.
2. Retrieve the public key of the CA.
3. Generate a membership certificate.

Class secmgr_tests::AJNCaTest_BasicTest_Test

Basic tests for the sample implementation of a CA based on AllJoyn.

1. Initialize an AJNCA instance.
2. Retrieve its public and private key.
3. Create another instance with the same storeName.
4. Check whether its private and public key are the same as the original AJNCA store.
5. Creating an AJNCA with an empty storeName should fail.
6. Reset the AJNCA store.
7. Resetting an AJNCA twice should fail.

Class secmgr_tests::ApplicationUpdaterTests_InstallIdentity_Test

Update the identity certificate for an offline application and check whether it was successfully updated when it comes back online.

1. Claim and stop remote application.
2. Update the stored identity certificate of the application.
3. Restart the remote application.
4. Wait for the updates to complete.
5. Ensure the identity certificate is correctly installed.

Class secmgr_tests::ApplicationUpdaterTests_InstallMembership_Test

Install a membership certificate for an offline application and check whether it was successfully installed when it comes back online.

1. Claim and stop remote application.
2. Store a membership certificate for the application.
3. Restart the remote application.
4. Wait for the updates to complete.
5. Ensure the membership certificate is correctly installed.

Class secmgr_tests::ApplicationUpdaterTests_Reset_Test

Reset an offline application and check its claimable state when it comes back online.

1. Claim and stop a remote application.
2. Reset the application from storage.
3. Restart the remote application.
4. Wait for the updates to complete.
5. Check whether the remote application is CLAIMABLE.

Class secmgr_tests::ApplicationUpdaterTests_ResetPolicy_Test

Reset a policy for an offline application and check whether it was successfully reset when it comes back online.

1. Claim and install a policy on the application.
2. Stop remote application.
3. Reset the stored policy of the application.
4. Restart the remote application.
5. Wait for the updates to complete.
6. Ensure the policy is correctly reset.

Class secmgr_tests::ApplicationUpdaterTests_SyncErIdentity_Test

Update the identity certificate of an application with an invalid certificate, and check whether a sync error of type SYNC_ER_POLICY is triggered.

1. Claim the remote application and stop it.
2. Make sure CAStorage returns an invalid certificate (e.g. digest mismatching the associated manifest).
3. Restart the remote application.
4. Check if a sync error of type SYNC_ER_IDENTITY is triggered.

Class secmgr_tests::ApplicationUpdaterTests_SyncErMembership_Test

Install a membership certificate of an application with an invalid certificate, and check whether a sync error of type SYNC_ER_MEMBERSHIP is triggered.

1. Claim the remote application and stop it.
2. Make sure CAStorage returns an invalid certificate (e.g. empty guid).
3. Restart the remote application.
4. Check if a sync error of type SYNC_ER_MEMBERSHIP is triggered.

Class secmgr_tests::ApplicationUpdaterTests_SyncErPolicy_Test

Install a permission policy with an older version than the one currently installed, and check if a sync error of type SYNC_ER_POLICY is triggered.

1. Claim the remote application and stop it.
2. Update the policy of the application to a previous version.
3. Restart the remote application.
4. Check if a sync error of type SYNC_ER_POLICY is triggered.

Class secmgr_tests::ApplicationUpdaterTests_SyncErReset_Test

Make sure resetting of an application fails, and check if a sync error of type SYNC_ER_RESET is triggered.

1. Claim the remote application.
2. Update the policy with a different admin group.
3. Stop the remote application.
4. Remove the application from the CA.
5. Restart the remote application.
6. Check if a sync error of type SYNC_ER_RESET is triggered.

Class secmgr_tests::ApplicationUpdaterTests_SyncErStorage_Test

Stop the CAStorage, and make sure the application updater starts notifying its listeners of some SYNC_ER_STORAGE errors when updating an application.

1. Claim the remote application and stop it.
2. Stop the CAStorage layer (or make sure that some basic functions like GetManagedApplication start returning errors).
3. Restart the remote application.
4. Check if a sync error of type SYNC_ER_STORAGE is triggered.

Class secmgr_tests::ApplicationUpdaterTests_UpdateAll_Test

Change the complete security configuration of an offline application and check whether it was successfully updated when it comes back online.

1. Claim and stop remote application.
2. Store a membership certificate for the application.
3. Update stored policy for the application.
4. Update stored identity certificate for the application.
5. Restart the remote application.
6. Wait for the updates to complete.
7. Ensure the membership certificate is installed correctly.
8. Ensure the policy is updated correctly.
9. Ensure the identity certificate is updated correctly.
10. Stop the remote application again.
11. Reset the remote application from storage.
12. Restart the remote application.
13. Check whether the remote application is CLAIMABLE again.

Class secmgr_tests::ApplicationUpdaterTests_UpdatePolicy_Test

Update a policy for an offline application and check whether it was successfully updated when it comes back online.

1. Claim and stop remote application.
2. Update the stored policy of the application.
3. Restart the remote application.
4. Wait for the updates to complete.
5. Ensure the policy is correctly installed.

Class secmgr_tests::CertChainHandlingTests_ClaimChain_Test

Claim an application by presenting the agent an identity certificate chain.

1. Claim an application and provide the agent an identity certificate chain
2. Check whether the application is CLAIMED.
3. Check if the application returns the full identity certificate chain

Class secmgr_tests::CertChainHandlingTests_InstallMembershipChain_Test

Install the membership certificates by presenting the agent membership certificate chains.

1. Install a membership certificate chain on application and provide the agent a membership certificate chain
2. Check whether the application is updated successfully.
3. Check if the application returns the full membership certificate chain
4. Install 2 more chains and verify they are found as well
5. Check if these chains can be removed and the application presents an empty list of membership certificates

Class secmgr_tests::CertChainHandlingTests_RegisterAgent_Test

Validate that the register agent is able to handle a identity certificate chain and multiple membership certificate chains

1. update the identity of an application and provide the agent an identity certificate chain
2. Check if the agent returns a correct membership list
3. Check if the agent returns the full identity certificate chain

Class secmgr_tests::CertChainHandlingTests_UpdateIdentityChains_Test

Update the identity of an already claimed application by presenting the agent an identity certificate chain.

1. update the identity of an application and provide the agent an identity certificate chain
2. Check whether the application is updated successfully.
3. Check if the application returns the full identity certificate chain

Class secmgr_tests::CertificateUtilTests_FailedToMembershipAndIdentityCertificate_Test

Verify the creation on membership and identity certificates fails if wrong validity is provided.

1. Create an Application.
2. Create a GroupInfo groupInfo and IdentityInfo identityInfo.
3. Use ToMembershipCertificate with zero validity period and make sure it fails (!= ER_OK).
4. Use ToIdentityCertificate zero validity period and make sure it fails (!= ER_OK).

Class secmgr_tests::CertificateUtilTests_ToIdentityCertificate_Test

Verify the creation on identity certificate.

1. Create an Application with a valid full KeyInfoNISTP256 keyinfo.
2. Create a valid IdentityInfo identityInfo with a valid guid.
3. Declare a validityPeriod with a valid value.
4. Use ToIdentityCertificate and make sure it succeeds and returns an identityCert.
5. Verify that the identityCert fields are matching the ones passed on during creation.
6. Verify that identityCert is not a CA.
7. Verify that identityCert has no digest.
8. Verify that identityCert has a nullptr encoding.

Class secmgr_tests::CertificateUtilTests_ToMembershipCertificate_Test

Verify the creation on membership certificate.

1. Create an Application with a valid full KeyInfoNISTP256 keyinfo.
2. Create a valid GroupInfo groupInfo with a valid guid.
3. Declare a validityPeriod with a valid value.
4. Use ToMembershipCertificate and make sure it succeeds and returns a membershipCert.
5. Verify that the membershipCert fields are matching the ones passed on during creation and that it is not a CA.
6. Verify that membershipCert has a nullptr encoding.

Class secmgr_tests::ClaimContextTests_ApproveManifest_Test

Verify the ApproveManifest function is working as expected.

1. Create an ClaimContext object
2. Call the ApproveManifest function with different values and validate the result

Class secmgr_tests::ClaimContextTests_BasicConstructor_Test

Verify the construction of a ClaimContext and its getters.

1. Create an ClaimContext object
2. Call the Get functions and check the return values

Class secmgr_tests::ClaimContextTests_SetClaimType_Test

Verify the SetClaimType function is working as expected.

1. Create an ClaimContext object
2. Call the SetClaimType function with different values and validate the results

Class secmgr_tests::ClaimingTests_BasicRobustness_Test

Basic robustness tests for the claiming, including input validation, unavailability of manifest listener/CA.

1. Claiming an off-line/unknown application should fail.
2. Claiming using an unknown identity should fail.
3. Claiming an application that is NOT_CLAIMABALE should fail.
4. Claiming an application that is CLAIMED should fail.
5. Claiming an application that NEED_UPDATE should fail.

Class secmgr_tests::ClaimingTests_ClaimListenerErrors_Test

Verify when the ClaimListener returns errors, these are handled correctly.

1. Start an application and make sure it's in the CLAIMABLE state
2. Try to claim it and trigger error conditions in the ClaimListener.
3. Verify that the application remains CLAIMABLE.

Class secmgr_tests::ClaimingTests_ConcurrentClaimListenerUpdate_Test

Changing the manifest listener when being in the callback of the original manifest listener should work.

1. Claim a CLAIMABLE application with a known identity.
2. While the claim listener is called to approve the manifest, a new manifest listener is installed to reject the manifest.
3. The original listener accepts the manifest.
4. The application should be claimed.
5. Start a new application and try claiming it.
6. The manifest should be rejected and the claiming should fail.
7. Make sure the new application is still claimable.

Class secmgr_tests::ClaimingTests_ConcurrentClaimOfSameApp_Test

Verify that the agent rejects claim of an application when it is already claiming that application

1. Start an application and make sure it is claimable.
2. Claim it, but make sure to claim it again while the first claim is ongoing
3. Verify that the application becomes claimed and the second claim fails

Class secmgr_tests::ClaimingTests_ConcurrentPSKClaims_Test

Verify that the agent concurrently can claim multiple applications.

1. Start multiple applications and try to claim them in parallel
2. Verify that all applications become claimed.

Class secmgr_tests::ClaimingTests_NestedPSKClaims_Test

Verify when the ClaimListener claim another application, these extra claims are handled correctly.

1. Start multiple application and make sure it's in the CLAIMABLE state
2. Try to claim them nesting the claim calls
3. Verify that the applications are CLAIMED.

Class secmgr_tests::ClaimingTests_OOBFailedClaiming_Test

Verify claiming with Out-Of-Band (OOB) fails when wrong PSK is used.

1. Start an application and make sure it's in the CLAIMABLE state with PSK preference; i.e., OOB.
2. Make sure that the security agent has generated the PSK.
3. Verify the the application uses a different PSK for OOB claiming.

4. Verify that the application is still CLAIMABLE and online and that claiming has failed.
5. Repeat the scenario where the PSK is generated by the application instead of the security agent and make sure PSK claiming fails.

Class secmgr_tests::ClaimingTests_OOBSuccessfulClaiming_Test

Verify claiming with Out-Of-Band (OOB) succeeds.

1. Start an application and make sure it's in the CLAIMABLE state with PSK preference; i.e., OOB.
2. Make sure that the application has generated the PSK.
3. Verify the the security agent uses the same PSK for OOB claiming and accepts the manifest.
4. Verify that the application is CLAIMED and online.
5. Reset/remove the application and make sure it's claimable again and repeat the scenario.
6. Verify that claiming was successful and that the application is in CLAIMED state and online.

Class secmgr_tests::ClaimingTests_RecoveryFromClaimingFailure_Test

Recovery from failure of notifying the CA of failure of claiming an application should be graceful.

1. Start a test application.
2. Install a manifest listener that stops the application before accepting the manifest, which will make the claiming fail.
3. Make sure the UpdatesCompleted to storage fails.
4. Claim the application and check that this fails.
5. Wait for a sync error for the application.

Class secmgr_tests::ClaimingTests_RejectManifest_Test

Reject the manifest during claiming and check whether the application becomes CLAIMABLE again.

1. Claim the remote application.
2. Reject the manifest.
3. Check whether the agent returns an ER_MANIFEST_REJECTED error.
4. Check whether the application remains CLAIMABLE.

Class secmgr_tests::ClaimingTests_ResetAfterReportClaimFails_Test

Verify that the agent resets the application after it claims it, but receives an error from the storage

1. Start an application and make sure it is claimable.
2. Claim it, but make sure that the storage FinishApplicationClaiming fails
3. Verify that the application becomes claimed for a short while and then claimable again

Class secmgr_tests::ClaimingTests_SuccessfulClaim_Test

Claim an application and check that it becomes CLAIMED.

1. Start the application.
2. Make sure the application is in a CLAIMABLE state.
3. Claim the application.
4. Accept the manifest of the application.
5. Check whether the application becomes CLAIMED.

Class secmgr_tests::ConcurrentUpdateTests_InstallMembershipAfterPolicy_Test

Install a membership certificate on an application while updating its policy and check whether both are installed correctly.

1. Claim an application.
2. Update the policy of the application.
3. While updating the policy, store a membership certificate for the policy.
4. Wait for the updates to complete.

5. Check whether the policy was updated correctly.
6. Check whether the membership was installed correctly.

Class secmgr_tests::ConcurrentUpdateTests_ResetAfterPolicy_Test

Reset an application while updating its policy and check whether it is CLAIMABLE.

1. Claim an application.
2. Update the policy of the application.
3. While updating the policy, reset the application using the security agent.
4. Check whether the application is CLAIMABLE.

Class secmgr_tests::ConcurrentUpdateTests_UpdateMultiple_Test

Install a membership certificate on an application and update its policy while updating its policy, and check whether the last policy and the membership certificate have been installed successfully.

1. Claim an application.
2. Update the policy of the application.
3. While updating the policy, install a membership certificate.
4. While installing the membership certificate, update its policy.
5. Wait for the updates to complete.
6. Check whether the last policy is installed correctly.
7. Check whether the membership certificate was installed correctly.

Class secmgr_tests::ConcurrentUpdateTests_UpdatePolicyAfterPolicy_Test

Update the policy of an application while updating its policy and check whether the last policy is installed successfully.

1. Claim an application.
2. Update the policy of the application.
3. While updating the policy, store another policy for the application.
4. Wait for the updates to complete.
5. Check whether the last policy is installed correctly.

Class secmgr_tests::GroupManagementTests_AuthoritiesCheck_Test

Check whether more than one group authority can be supported.

1. Create a GroupInfo (group0) object with an empty authority
2. Store it successfully and make sure it gets an authority set.
3. Create another GroupInfo (group1) and assign it group0's GUID but a different authority.
4. Verify storing of group1 fails as it uses group0's GUID.
5. Change group1 to use the same authority as in group0.
6. Verify updating storage with group1 succeeds and that both groups are identical.
7. Create a different group2 with a different GUID and Authority.
8. Make sure its storage and retrieval succeed and that it is different than both previous groups.

Class secmgr_tests::GroupManagementTests_DefaultAuthority_Test

Check whether the default group authority is added on all Group methods.

1. Create a GroupInfo object.
2. Store the GroupInfo object and verify the authority is set.
3. Create another GroupInfo object and fill in only the guid.
4. Check if the original GroupInfo object can be retrieved.
5. Create another GroupInfo object and fill in only the guid.
6. Check if the original GroupInfo object can be removed.

Class secmgr_tests::GroupManagementTests_FailedBasicGroupOperations_Test

Retrieval and deletion of unknown groups should fail.

1. Try to get an unknown group and make sure this fails.
2. Try to remove an unknown group and make sure this fails.
3. Try to get all managed groups and make sure the vector is empty.

Class secmgr_tests::GroupManagementTests_GroupManipBasic_Test

Store, retrieve and delete a group from storage.

1. Define a valid security group and store it.
2. Retrieve the group from storage and check whether it matches the stored group.
3. Remove the security group.
4. Retrieving the security group should fail.

Class secmgr_tests::GroupManagementTests_GroupManipManyGroups_Test

Store, retrieve and delete many security groups from storage.

1. Define a series of security groups and store them one by one.
2. Retrieve all groups from storage and count whether all have been stored correctly.
3. Remove all groups one by one from storage.
4. Retrieve all groups from storage and make sure none are returned.

Class secmgr_tests::GroupManagementTests_GroupUpdate_Test

Update an existing group and make sure it can be retrieved correctly.

1. Create a valid security group.
2. Store the group and make sure this is successful.
3. Retrieve the group from storage and make sure this is successful.
4. Change the name and description of the group.
5. Store the group and make sure this is successful.
6. Retrieve the group and make sure it matches the updated info.

Class secmgr_tests::IdentityManagementTests_FailedBasicIdentityOperations_Test

Retrieval and deletion of unknown identities should fail.

1. Try to get an unknown identity and make sure this fails.
2. Try to remove an unknown identity and make sure this fails.
3. Try to get all identities and make sure there are none.

Class secmgr_tests::IdentityManagementTests_IdentityManipBasic_Test

Store, retrieve and delete an identity from storage.

1. Define a valid IdentityInfo and store it.
2. Retrieve the identity from storage and check whether it matches the stored identity.
3. Remove the identity from storage.
4. Retrieving the identity should fail.

Class secmgr_tests::IdentityManagementTests_IdentityManipManyIdentities_Test

Store, retrieve and delete many identities from storage.

1. Define a series of identities and store them one by one.
2. Retrieve all identities from storage and count whether all have been stored correctly.
3. Remove all identities one by one from storage.
4. Retrieve all identities from storage and make sure none are returned.

Class secmgr_tests::IdentityManagementTests_IdentityUpdate_Test

Update an existing identity and make sure it can be retrieved correctly.

1. Create a valid identity.
2. Store the identity and make sure this is successful.
3. Retrieve the identity from storage and make sure this is successful.
4. Change the name and description of the identity.
5. Store the identity and make sure this is successful.
6. Retrieve the identity and make sure it matches the updated identity.

Class secmgr_tests::IdentityTests_SuccessfulInstallIdentity_Test

Update the identity certificate of an application and check that it gets installed correctly.

1. Start the application.
2. Make sure the application is in a CLAIMABLE state.
3. Create and store an IdentityInfo.
4. Claim the application using the IdentityInfo.
5. Check whether the application becomes CLAIMED.
6. Create and store another IdentityInfo.
7. Update the identity certificate of the application.
8. Wait for the updates to be completed.
9. Check whether the identity certificate was installed successfully.
10. Remove the latest identity and make sure the app is removed and that it becomes claimable again.
11. Use the original identity to claim 2 apps successfully.
12. Remove the original identity and verify that the applications are removed and they are claimable again.
13. Get all managed applications and verify that none exists.

Class secmgr_tests::IdentityTests_UpdateIdentityPolicyUpdate_Test

Verify that update identity triggers an increase of the policy version

1. Start an application and make sure it's online and CLAIMABLE.
2. Successfully store an IdentityInfo instance.
3. Successfully claim the application using the IdentityInfo instance.
4. Make sure the application is online and in the CLAIMED state with no updates pending.
5. Make sure the remote identity and manifest of the application match the stored ones.
6. Update a policy on an application
7. Verify that updateIdentity is succesful.
8. Make sure updates have been completed.
9. Check that the policy version increased.

Class secmgr_tests::ManifestTests_UpdateManifest_Test

Update the manifest of an application and check whether a ManifestUpdate event is triggered if the manifest contains additional rules.

1. Set the manifest of the application to manifest1.
2. Claim the application and check whether the manifest during claiming matches the remote manifest.
3. Set the manifest of the application to manifest2 which extends manifest1.
4. Check whether a ManifestUpdate event is triggered.
5. Update the identity certificate based on the newly requested manifest.
6. Check that no additional ManifestUpdate events are triggered.
7. Set the manifest of the application to manifest1.

8. Make sure no additional ManifestUpdate events are triggered.

Class secmgr_tests::ManifestUtilTests_Difference_Test

Verify the difference between two manifest is computed correctly.

1. Create two manifests: one basic manifest, and one extending the basic manifest by adding another interface and by extending the action mask on a specific member.
2. Compute the difference between the extended manifest and the basic manifest, and check whether the outcome is as expected.
3. Compute the difference between the basic manifest and the extend manifest, and make sure it is empty.

Class secmgr_tests::ManifestUtilTests_ExtendedPermissionPolicyDigest_Test

Verify the PermissionPolicy's digest consistency after copy or assignment operations. Also verify the functionalities provided by the static public Util class.

1. Using a DefaultPolicyMarshaller create a PermissionPolicy (permPolicy) and get its digest.
2. Successfully create a copy PermissionPolicy (permPolicyCopy) and get its digest.
3. Compare digests from permPolicyCopy and permPolicy and make sure they match.
4. Create a PermissionPolicy (permPolicyAssignee) using the assignment operator from permPolicy and make sure its digest matches those from permPolicyCopy and permPolicy.
5. In a scoped block, initialize the Util class and get the byte-array of permPolicy successfully. Using that byteArray invoke GetPolicy on Util to create a policyFromImport successfully. Finalize the Util successfully.
6. Finally, make sure the digest of policyFromImport matches that of permPolicy.

Class secmgr_tests::ManifestUtilTests_ManifestConstruction_Test

Verify the construction of valid Manifest objects using the Manifest class. Also verify the provided operators and the digest matching.

1. Create an empty manifest object and make sure that its rules and byte-array are empty. Both sizes of rules and the byte-array should be zero.
2. Create a manifest from some two generated rules (manifestFromRules) and verify that it has those exact generated rules. Also, make sure that the corresponding byte-array is not empty.
3. Repeat the previous step for a manifest created form the previous byte-array (manifestFromByteArray). Also, verify that the byte-array of manifestFromRules matches that of manifestFromByteArray.
4. Get digests of both manifestFromByteArray and manifestFromRules and make sure they match.
5. Create a copy (copyManifest) form manifestFromByteArray using the copy constructor and make sure == and != operators against manifestFromByteArray and manifestFromRules hold.
6. Create a manifestAssignee using the assignment operator form manifestFromByteArray and make sure == and != operators against manifestFromByteArray and manifestFromRules hold.
7. Get the digests from manifestAssignee, manifestFromByteArray and copyManifest and make sure they are all identical.

Class secmgr_tests::ManifestUtilTests_ManifestIllegalArgs_Test

Verify that the Manifest class methods can handle illegal arguments.

1. Try to construct a manifest from nullptr rules and/or a size <= 0 and make sure then it is equal to a manifest created with the default constructor.
2. Try to construct a manifest from nullptr byte-array and/or a size <= 0 and make sure then it is equal to a manifest created with the default constructor.
3. Call all getter functions on a manifest created with the default constructor with nullptr inputs and make sure this fails (!= ER_OK).
4. Call all setter functions on a manifest created with the default constructor with nullptr first argument and size <= 0 and make sure all will fail (!= ER_OK).
5. Call all setter functions on a manifest created with the default constructor with valid first argument and size <= 0 and make sure all will fail (!= ER_OK).

Class secmgr_tests::ManifestUtilTests_UtilIllegalArgs_Test

Verify that the Util class methods can handle illegal arguments.

1. Init Util with a nullptr argument and make sure it fails.
2. Call all methods of Util with valid arguments and make sure they all fail (!= ER_OK).
3. Init Util with a valid busattachment and make sure it succeeds.
4. Call GetDefaultMarshaller with nullptr argument and make sure it fails (!= ER_OK).
5. Call GetPolicyByteArray with nullptr byte-array and size and make sure this fails (!= ER_OK).
6. Call GetPolicy with nullptr byte-array and/or size <= 0 and make sure this fails (!= OR_OK).
7. Call Fini on Util and make sure it succeeds (== ER_OK).

Class secmgr_tests::MembershipTests_InstallRemoveMembershipPolicyUpdate_Test

Verify that installing and removing a membership triggers an increase in the policy version

1. Start an application and make sure it's online and CLAIMABLE.
2. Successfully store an IdentityInfo instance.
3. Successfully claim the application using the IdentityInfo instance.
4. Make sure the application is online and in the CLAIMED state with no updates pending.
5. Make sure the remote identity and manifest of the application match the stored ones.
6. Update a policy on an application
7. Verify that installing of membership using groupInfo1 is successful.
8. Make sure updates have been completed.
9. Check that the policy version increased.
10. Verify that removing of membership using groupInfo1 is successful.
11. Make sure updates have been completed.
12. Check that the policy version increased again.

Class secmgr_tests::MembershipTests_SuccessfulInstallMembership_Test

Verify the ability to install several memberships based on different GroupInfo instances.

1. Store a couple of different GroupInfo instances; groupInfo1 and groupInfo2 in persistency.
2. Start an application and make sure it's online and CLAIMABLE.
3. Try to install and remove a membership using the lately announced application and make sure this fails.
4. Successfully store an IdentityInfo instance.
5. Successfully claim the application using the IdentityInfo instance.
6. Make sure the application is online and in the CLAIMED state with no updates pending.
7. Make sure the remote identity and manifest of the application match the stored ones.
8. Verify that installing of membership using groupInfo1 is successful.
9. Make sure updates have been completed.
10. Repeat the previous 2 steps for groupInfo2.
11. Verify that removal of membership using groupInfo1 is successful.
12. Make sure updates have been completed.
13. Repeat the previous 2 steps for groupInfo2.
14. Install memberships for both groupInfo1 and groupInfo2 successfully.
15. Verify that deleting groupInfo1 and groupInfo2 will result in syncing the app again and in removing the memberships associated.
16. Repeat the previous step but verify the removal of memberships associated immediately after the deletion of each group.

Class secmgr_tests::MultiAgentAppTests_DISABLED_SuccessfulClaimAndUnclaimManyApps_Test

Verify that where multi-agents are active, the applications needing to be claimed and be managed are dealt with consistently.

1. Start an number of security agents which share the same storage.
2. Start a number of applications and make sure that all agents see those applications in the CLAIMABLE state.
3. Randomly, let the security agents start claiming random claimable applications.
4. Verify that all applications have been claimed successfully.
5. Randomly, let the security agents start unclaiming random claimed applications.
6. Verify that all applications have changed state to CLAIMABLE

Class secmgr_tests::MultiAgentAppTests_DISABLED_SuccessfulRestartSecAgentWithManyApps_Test

Verify that a restarted Security Agent (SA) after claiming many applications will maintain a consistent view.

1. Start a number of applications and make sure they are claimable.
2. Start an SA and verify that it could see all the online applications in a CLAIMABLE state.
3. Ask the SA to start claiming (at no specific order) all claimable applications.
4. Verify that all applications have been claimed; state change to CLAIMED.
5. Ask the SA to install/update all claimed applications with a predefined membership certificate and policy.
6. Kill the SA.
7. Start the SA again and make sure that all previously claimed applications are in a consistent online state and no sync errors will have occurred.
8. Kill all applications.
9. Verify that the SA considers all applications as offline.

Class secmgr_tests::MultiAgentAppTests_DISABLED_SuccessfulUpdateManyApps_Test

Verify that where multi-agents are active, claimed applications can be updated correctly after a restart.

1. Start an number of security agents which share the same storage.
2. Start a number of applications and make sure that all agents see those applications in the CLAIMABLE state.
3. Claim successfully all applications...could be using one security agent.
4. Verify that all applications have been claimed successfully.
5. Stop all claimed applications.
6. Make sure all security agents see all applications as off-line (i.e., no busName).
7. Update all applications with predefined membership and policy using the available security agents randomly.
8. Verify that storage has the predefined membership and policy for all applications.
9. Restart all the applications.
10. Verify that all applications have been updated correctly and that no sync errors have been encountered.
11. Verify all applications are in the CLAIMED state.

Class secmgr_tests::PolicyGeneratorTest_BasicIllegalArgTest_Test

Basic test for illegal argument in policy generator.

1. Create an empty vector of GroupInfo; groupInfoEmptyVec
2. Use the existing PolicyGenerator instance (pg) to get a DefaultPolicy using the groupInfoEmptyVec and make sure this does not fail but return a default policy with one admin rule.

Class secmgr_tests::PolicyGeneratorTest_BasicTest_Test

Basic test for the sample policy generator.

1. Create a security group and store it.

2. Generate a policy for this group.
3. Make sure this policy has two ACLs (including one for the admin group).
4. Create another security group and store it.
5. Generate another policy for both groups.
6. Make sure this policy has three ACLs (including one for the admin group).

Class secmgr_tests::PolicyGeneratorTest_DenyRules_Test

Validate the generation of a policy with deny rules.

1. Create a policy generator and add a random application to its blacklist.
2. Generate a default policy for an empty vector of groups.
3. Check that the resulting policy has 2 ACLs.
4. Check that the resulting policy only contains valid deny rules.

Class secmgr_tests::PolicyTests_PermissionDenied_Test

Verify the the security agent can handle permission denied response.

1. Start the application and make sure it's claimable.
2. Claim the application successfully.
3. Install a policy that does NOT contain the admin group rule.
4. Check whether the application is in SYNC_PENDING state.
5. Make sure that at least one sync error is triggered.

Class secmgr_tests::PolicyTests_SuccessfulInstallPolicyAndUpdatePolicy_Test

Update the policy of an application and check whether it is updated correctly.

1. Start the application.
2. Installing and retrieving the policy before claiming should fail.
3. Make sure the application is in a CLAIMABLE state.
4. Create and store an IdentityInfo.
5. Claim the application using the IdentityInfo.
6. Accept the manifest of the application.
7. Check whether the application becomes CLAIMED.
8. Make sure the retrieval of the policy returns ER_END_OF_DATA.
9. Update the policy.
10. Wait for updates to complete.
11. Update the policy again.
12. Check whether the remote policy is equal to the installed policy.
13. Check whether the remote policy is equal to the policy that can be retrieved from storage.
14. Wait for updates to complete.
15. Check whether the remote policy is equal to the installed policy.
16. Check whether the remote policy is equal to the policy that can be retrieved from storage.
17. Try to install a newer policy (version 100) and verify it was successful
18. Try to install an older policy (version 1) and verify this fails
19. Try to install a default policy of (version 0) and verify this was successful.
20. Get the persisted policy and make sure its version is (version 100 +1)

Class secmgr_tests::PolicyTests_SuccessfulResetPolicy_Test

Verify resetting the policy of an application succeeds.

1. Start the application and make sure it's claimable.
2. Claim the application successfully.

3. Check the default policy.
4. Install a different policy and wait until updates have been completed.
5. Check whether the policy was installed successfully.
6. Reset the policy and wait until updates have been completed.
7. Check the default policy.

Class secmgr_tests::PolicyUtilTest_NormalizePolicy_Test

Normalize a policy with partially matching rules, and check whether the resulting policy matches the expected outcome.

1. Create a rule with a specific InterfaceName, and one Members with a specific MemberName, and with the ActionMask set to ACTION_OBSERVE.
2. Create another rule with the same InterfaceName, and two Members. One member with a matching name to the previous rule, and one with a different name, each having the ActionMask set to ACTION_MODIFY.
3. Create a third rule with a different InterfaceName but with a member of a matching MemberName. The ActionMask should be set to ACTION_PROVIDE.
   (a) Add those rules to a policy.
   (b) Normalize the policy.
4. Verify that the normalized policy has two rules (one for each InterfaceName).
5. Verify that the rule with the matching InterfaceName contains two members and verify that the ActionMasks are collapsed successfully.
6. Verify that the rule with the unique InterfaceName has only one member and verify the resulting ActionMask.

Class secmgr_tests::PolicyUtilTest_NormalizePolicyMembers_Test

Normalize a rule with matching members, and see whether the resulting action mask corresponds to the OR value of the ActionMasks of all individual members.

1. Create a rule with three matching members with different action masks: one with ACTION_PROVIDE, one with ACTION_OBSERVE and one with ACTION_MODIFY.
2. Add this rule to a policy.
3. Normalize this policy.
4. Verify that the normalized policy has only one member.
5. Verify that this member has a full ActionMask.

Class secmgr_tests::PolicyUtilTest_NormalizePolicyRules_Test

Normalize a policy with matching rules and members, and see whether the resulting action mask corresponds to the OR of the ActionMasks of all members.

1. Create a rule with two matching members with different action masks: one with ACTION_OBSERVE and one with ACTION_MODIFY.
2. Create another rule that matches the previous members, but with a different action mask ACTION_PROVIDE.
3. Add those rules to a policy.
4. Normalize this policy.
5. Verify that the normalized policy has only one rule.
6. Verify that the member of this rule has a full ActionMask.

Class secmgr_tests::ResetTests_RecoveryFromRemoteReset_Test

Discovery of an application that has been remotely reset, should result in a sync error and should be reclaimable after removing it from storage.

1. Start a test application and claim it.
2. Reset the application behind the back of the security manager.
3. Check that the security manager reports a SYNC_ER_UNEXPECTED_STATE.

4. Check that reclaiming the application would fail.
5. Forcibly remove the application from storage.
6. Check that reclaiming the application now succeeds.

Class secmgr_tests::ResetTests_RecoveryFromResetFailure_Test

Recovery from failure of notifying the CA of failure to reset an application should be graceful.

1. Start a test application and claim it.
2. Make sure remote reset fails.
3. Stop the application.
4. Make sure the UpdatesCompleted to storage fails.
5. Reset the application and check that this succeeds.
6. Restart the test application and make sure it is removed from storage.

Class secmgr_tests::ResetTests_RecoveryFromResetSuccess_Test

Recovery from failure of notifying the CA of successful resetting an application should be graceful.

1. Start a test application and claim it.
2. Make sure the UpdatesCompleted to storage fails.
3. Reset the application and check that this succeeds.
4. Restart the test application and make sure it is removed from storage.

Class secmgr_tests::ResetTests_SuccessfulReset_Test

Reset an application and make sure it becomes CLAIMABLE again.

1. Start the application.
2. Make sure the application is in a CLAIMABLE state.
3. Create and store an IdentityInfo.
4. Claim the application using the IdentityInfo.
5. Accept the manifest of the application.
6. Check whether the application becomes CLAIMED.
7. Remove the application from storage.
8. Check whether it becomes CLAIMABLE again.
9. Claim the application again.
10. Check whether it becomes CLAIMED again.

Class secmgr_tests::RestartAgentTests_SuccessfulAgentRestart_Test

Verify that the agent can restart and maintain a consistent view on the online applications. // See AS-1634 for the number of applications issue

1. Start an even X number of applications and make sure they are claimable.
2. Using a dedicated IdtityInfo (per application) and a default manifest, claim an X/2 (claimedApps) of the applications.
3. Verify that claimedApps are in a CLAIMED state and that the other X/2 (claimableApps) are still in a CLAIMABLE state.
4. Delete the security agent instance that claimed the claimedApps.
5. Create a new security agent which uses the same keystore and CAStorage.
6. Verify that all online applications are in a consistent online application state from the security agent perspective.

Class secmgr_tests::SecurityAgentFactoryTests_Basic_Test

Ensure that the security agent factory can return a valid security agent.

1. Get a security agent factory instance.

2. Get a security agent using a connected bus attachment.
3. Validate the security agent returned
4. Get a security agent with using a null bus attachment parameter.
5. Validate the security agent returned.

Class secmgr_tests::UIStorageTests_SetMetaData_Test

Set the user defined name of an application and check whether it can be retrieved.

1. Claim the remote application.
2. Set some meta data.
3. Retrieve the application from the security agent.
4. Check whether the retrieved meta data matches the data that was set.

Class secmgr_tests::UIStorageTests_StorageReset_Test

Ensure that resetting the database will trigger OnStorageReset.

1. Register a storage listener.
2. Reset the storage and make sure that OnStorageReset was called.
3. Unregister the listener.

Class secmgr_tests::UpdateFromSecmgrTest_BasicUpdateFromSecMgr_Test

Ensure that an explicit update applications call from the security agent will trigger the correct logic in the application updater.

1. Claim an application.
2. Trigger update applications from the security agent for all apps.
3. Ensure that the number of times update started and completed is correct.
4. Trigger update applications from the security agent using a vector containing the claimed app.
5. Ensure that the number of times update started and completed have been incremented.

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

File Index

4.1 File List

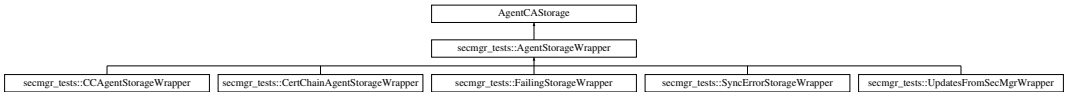
Here is a list of all documented files with brief descriptions:

Chapter 5

Class Documentation

5.1 secmgr_tests::AgentStorageWrapper Class Reference

Inheritance diagram for secmgr_tests::AgentStorageWrapper:



Public Member Functions

- **AgentStorageWrapper** (shared_ptr< AgentCAStorage > &_ca)
- virtual QStatus **GetManagedApplication** (Application &app) const
- virtual QStatus **RegisterAgent** (const KeyInfoNISTP256 &agentKey, const Manifest &manifest, GroupInfo &adminGroup, IdentityCertificateChain &identityCertificates, vector< MembershipCertificateChain > &adminGroupMemberships)
- virtual QStatus **StartApplicationClaiming** (const Application &app, const IdentityInfo &idInfo, const Manifest &manifest, GroupInfo &adminGroup, IdentityCertificateChain &idCert)
- virtual QStatus **FinishApplicationClaiming** (const Application &app, QStatus status)
- virtual QStatus **UpdatesCompleted** (Application &app, uint64_t &updateID)
- virtual QStatus **StartUpdates** (Application &app, uint64_t &updateID)
- virtual QStatus **GetCaPublicKeyInfo** (KeyInfoNISTP256 &keyInfoOfCA) const
- virtual QStatus **GetAdminGroup** (GroupInfo &groupInfo) const
- virtual QStatus **GetMembershipCertificates** (const Application &app, vector< MembershipCertificateChain > &membershipCertificates) const
- virtual QStatus **GetIdentityCertificatesAndManifest** (const Application &app, IdentityCertificateChain &identityCertificates, Manifest &manifest) const
- virtual QStatus **GetPolicy** (const Application &app, PermissionPolicy &policy) const
- virtual void **RegisterStorageListener** (StorageListener *listener)
- virtual void **UnRegisterStorageListener** (StorageListener *listener)

Protected Attributes

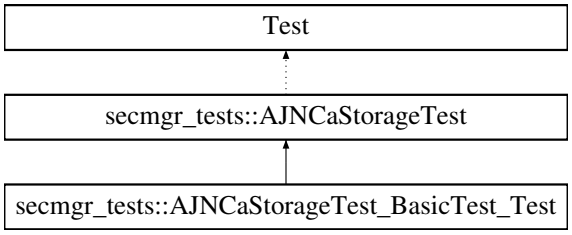
- shared_ptr< AgentCAStorage > **ca**

The documentation for this class was generated from the following file:

- AgentStorageWrapper.h

5.2 secmgr_tests::AJNCaStorageTest Class Reference

Inheritance diagram for secmgr_tests::AJNCaStorageTest:



Public Member Functions

- void **SetUp** ()
- void **TearDown** ()

Public Attributes

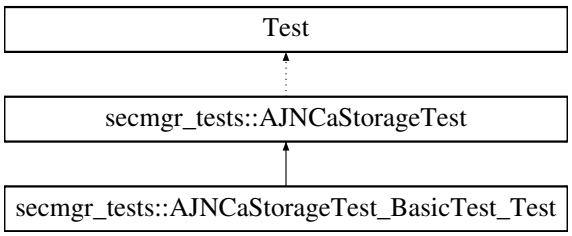
- shared_ptr< AJNCaStorage > **ca**
- shared_ptr< SQLStorage > **sql**

The documentation for this class was generated from the following file:

- AJNCaStorageTests.cc

5.3 secmgr_tests::AJNCaStorageTest_BasicTest_Test Class Reference

Inheritance diagram for secmgr_tests::AJNCaStorageTest_BasicTest_Test:



Additional Inherited Members

5.3.1 Detailed Description

Test Basic tests for the sample implementation of a CAStorage based on AllJoyn.

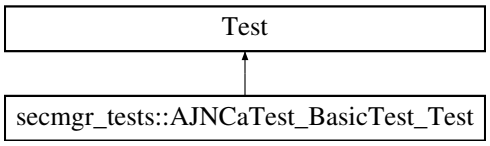
1. Initialize an AJNCAStorage.
2. Retrieve the public key of the CA.
3. Generate a membership certificate.

The documentation for this class was generated from the following file:

- AJNCaStorageTests.cc

5.4 secmgr_tests::AJNCaTest_BasicTest_Test Class Reference

Inheritance diagram for secmgr_tests::AJNCaTest_BasicTest_Test:



5.4.1 Detailed Description

Test Basic tests for the sample implementation of a CA based on AllJoyn.

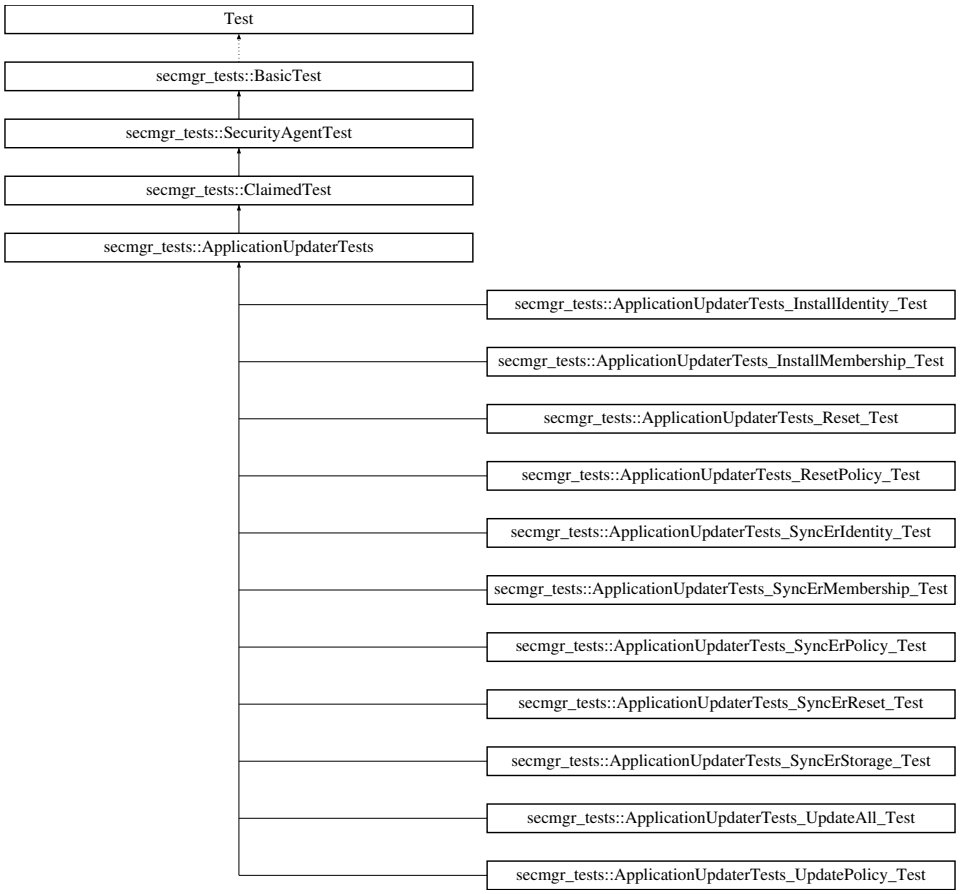
1. Initialize an AJNCA instance.
2. Retrieve its public and private key.
3. Create another instance with the same storeName.
4. Check whether its private and public key are the same as the original AJNCA store.
5. Creating an AJNCA with an empty storeName should fail.
6. Reset the AJNCA store.
7. Resetting an AJNCA twice should fail.

The documentation for this class was generated from the following file:

- AJNCaTests.cc

5.5 secmgr_tests::ApplicationUpdaterTests Class Reference

Inheritance diagram for secmgr_tests::ApplicationUpdaterTests:



Public Member Functions

- shared_ptr< AgentCAStorage > & **GetAgentCAStorage** ()

Public Attributes

- GroupInfo **groupInfo**
- PermissionPolicy **policy**
- vector< GUID128 > **policyGroups**
- shared_ptr
  < SyncErrorStorageWrapper > **wrappedCA**

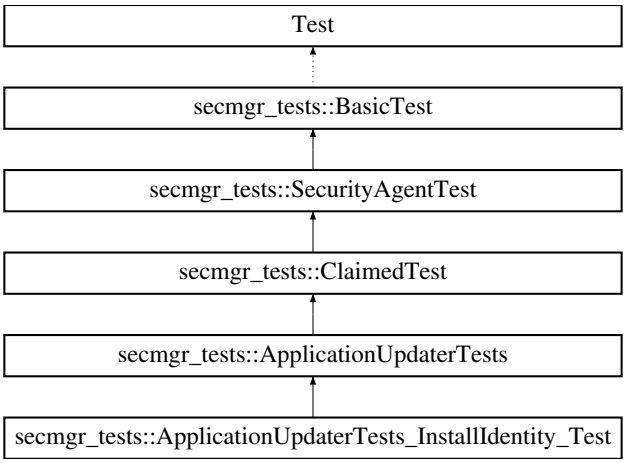
Additional Inherited Members

The documentation for this class was generated from the following file:

- ApplicationUpdaterTests.cc

5.6 secmgr_tests::ApplicationUpdaterTests_InstallIdentity_Test Class Reference

Inheritance diagram for secmgr_tests::ApplicationUpdaterTests_InstallIdentity_Test:



Additional Inherited Members

5.6.1 Detailed Description

Test Update the identity certificate for an offline application and check whether it was successfully updated when it comes back online.

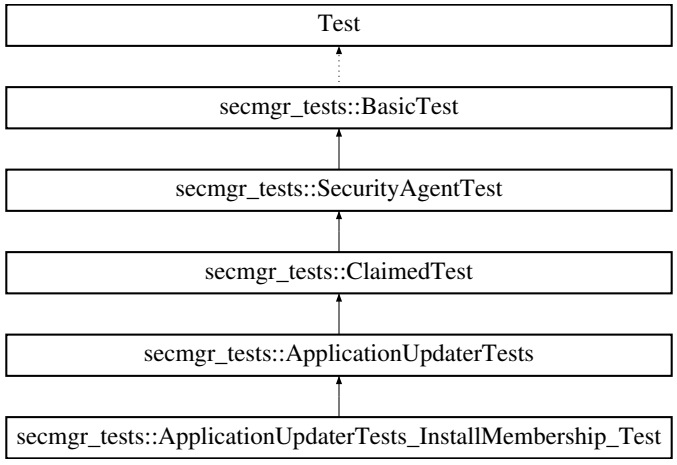
1. Claim and stop remote application.
2. Update the stored identity certificate of the application.
3. Restart the remote application.
4. Wait for the updates to complete.
5. Ensure the identity certificate is correctly installed.

The documentation for this class was generated from the following file:

- ApplicationUpdaterTests.cc

5.7 secmgr_tests::ApplicationUpdaterTests_InstallMembership_Test Class Reference

Inheritance diagram for secmgr_tests::ApplicationUpdaterTests_InstallMembership_Test:



Additional Inherited Members

5.7.1 Detailed Description

Test Install a membership certificate for an offline application and check whether it was successfully installed when it comes back online.

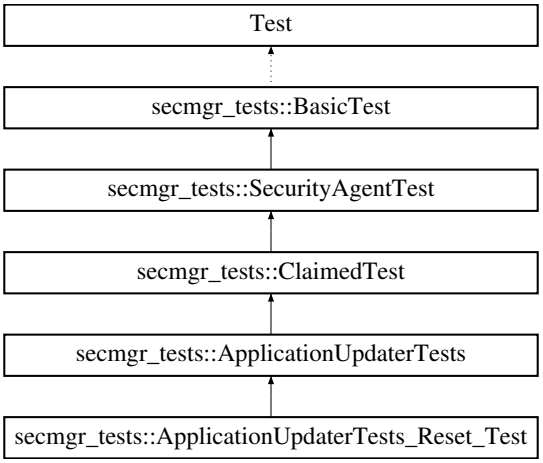
1. Claim and stop remote application.
2. Store a membership certificate for the application.
3. Restart the remote application.
4. Wait for the updates to complete.
5. Ensure the membership certificate is correctly installed.

The documentation for this class was generated from the following file:

- ApplicationUpdaterTests.cc

5.8 secmgr_tests::ApplicationUpdaterTests_Reset_Test Class Reference

Inheritance diagram for secmgr_tests::ApplicationUpdaterTests_Reset_Test:



Additional Inherited Members

5.8.1 Detailed Description

Test Reset an offline application and check its claimable state when it comes back online.

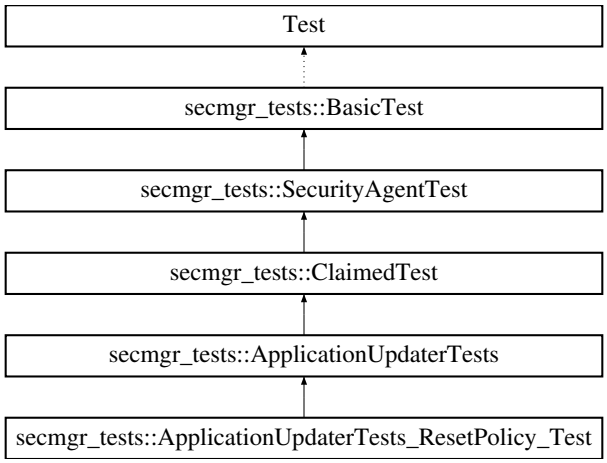
1. Claim and stop a remote application.
2. Reset the application from storage.
3. Restart the remote application.
4. Wait for the updates to complete.
5. Check whether the remote application is CLAIMABLE.

The documentation for this class was generated from the following file:

- ApplicationUpdaterTests.cc

5.9 secmgr_tests::ApplicationUpdaterTests_ResetPolicy_Test Class Reference

Inheritance diagram for secmgr_tests::ApplicationUpdaterTests_ResetPolicy_Test:



Additional Inherited Members

5.9.1 Detailed Description

Test Reset a policy for an offline application and check whether it was successfully reset when it comes back online.

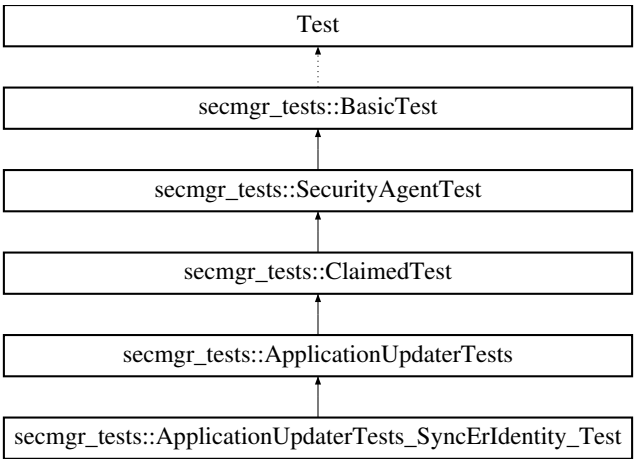
1. Claim and install a policy on the application.
2. Stop remote application.
3. Reset the stored policy of the application.
4. Restart the remote application.
5. Wait for the updates to complete.
6. Ensure the policy is correctly reset.

The documentation for this class was generated from the following file:

- ApplicationUpdaterTests.cc

5.10 secmgr_tests::ApplicationUpdaterTests_SyncErIdentity_Test Class Reference

Inheritance diagram for secmgr_tests::ApplicationUpdaterTests_SyncErIdentity_Test:



Additional Inherited Members

5.10.1 Detailed Description

Test Update the identity certificate of an application with an invalid certificate, and check whether a sync error of type SYNC_ER_POLICY is triggered.

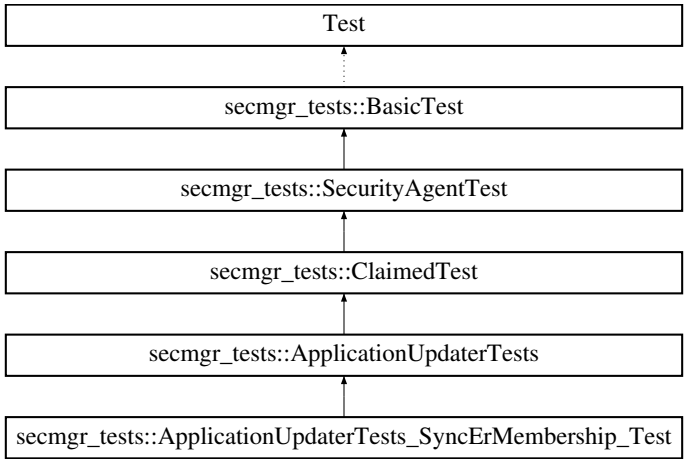
1. Claim the remote application and stop it.
2. Make sure CAStorage returns an invalid certificate (e.g. digest mismatching the associated manifest).
3. Restart the remote application.
4. Check if a sync error of type SYNC_ER_IDENTITY is triggered.

The documentation for this class was generated from the following file:

- ApplicationUpdaterTests.cc

5.11 secmgr_tests::ApplicationUpdaterTests_SyncErMembership_Test Class Reference

Inheritance diagram for secmgr_tests::ApplicationUpdaterTests_SyncErMembership_Test:



Additional Inherited Members

5.11.1 Detailed Description

Test Install a membership certificate of an application with an invalid certificate, and check whether a sync error of type SYNC_ER_MEMBERSHIP is triggered.

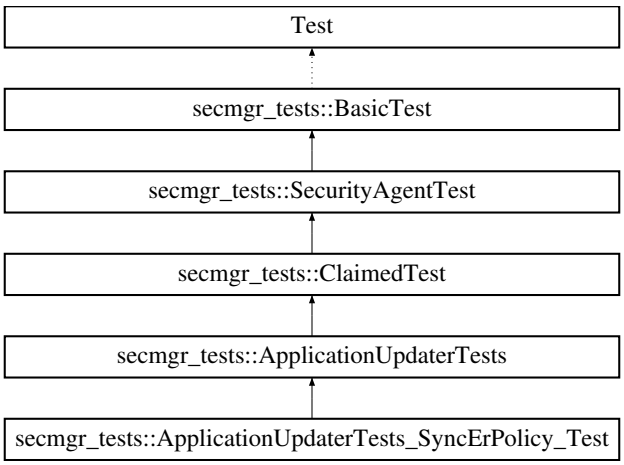
1. Claim the remote application and stop it.
2. Make sure CAStorage returns an invalid certificate (e.g. empty guid).
3. Restart the remote application.
4. Check if a sync error of type SYNC_ER_MEMBERSHIP is triggered.

The documentation for this class was generated from the following file:

- ApplicationUpdaterTests.cc

5.12 secmgr_tests::ApplicationUpdaterTests_SyncErPolicy_Test Class Reference

Inheritance diagram for secmgr_tests::ApplicationUpdaterTests_SyncErPolicy_Test:



Additional Inherited Members

5.12.1 Detailed Description

Test Install a permission policy with an older version than the one currently installed, and check if a sync error of type SYNC_ER_POLICY is triggered.

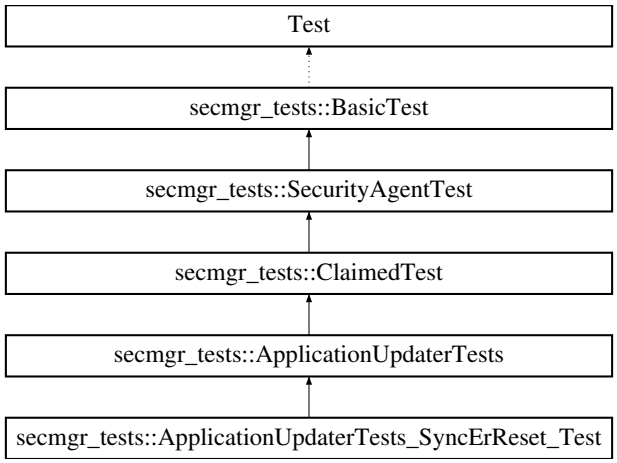
1. Claim the remote application and stop it.
2. Update the policy of the application to a previous version.
3. Restart the remote application.
4. Check if a sync error of type SYNC_ER_POLICY is triggered.

The documentation for this class was generated from the following file:

- ApplicationUpdaterTests.cc

5.13 secmgr_tests::ApplicationUpdaterTests_SyncErReset_Test Class Reference

Inheritance diagram for secmgr_tests::ApplicationUpdaterTests_SyncErReset_Test:



Additional Inherited Members

5.13.1 Detailed Description

Test Make sure resetting of an application fails, and check if a sync error of type SYNC_ER_RESET is triggered.

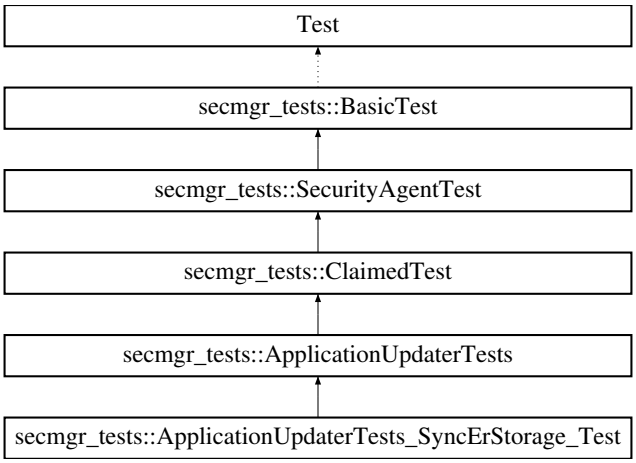
1. Claim the remote application.
2. Update the policy with a different admin group.
3. Stop the remote application.
4. Remove the application from the CA.
5. Restart the remote application.
6. Check if a sync error of type SYNC_ER_RESET is triggered.

The documentation for this class was generated from the following file:

- ApplicationUpdaterTests.cc

5.14 secmgr_tests::ApplicationUpdaterTests_SyncErStorage_Test Class Reference

Inheritance diagram for secmgr_tests::ApplicationUpdaterTests_SyncErStorage_Test:



Additional Inherited Members

5.14.1 Detailed Description

Test Stop the CAStorage, and make sure the application updater starts notifying its listeners of some SYNC_ER_STORAGE errors when updating an application.

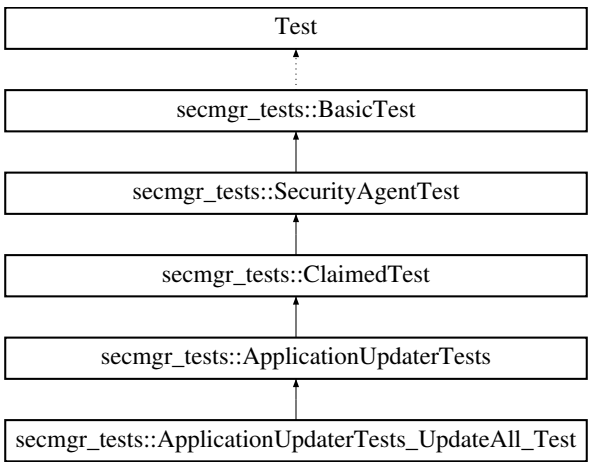
1. Claim the remote application and stop it.
2. Stop the CAStorage layer (or make sure that some basic functions like GetManagedApplication start returning errors).
3. Restart the remote application.
4. Check if a sync error of type SYNC_ER_STORAGE is triggered.

The documentation for this class was generated from the following file:

- ApplicationUpdaterTests.cc

5.15 secmgr_tests::ApplicationUpdaterTests_UpdateAll_Test Class Reference

Inheritance diagram for secmgr_tests::ApplicationUpdaterTests_UpdateAll_Test:



Additional Inherited Members

5.15.1 Detailed Description

Test Change the complete security configuration of an offline application and check whether it was successfully updated when it comes back online.

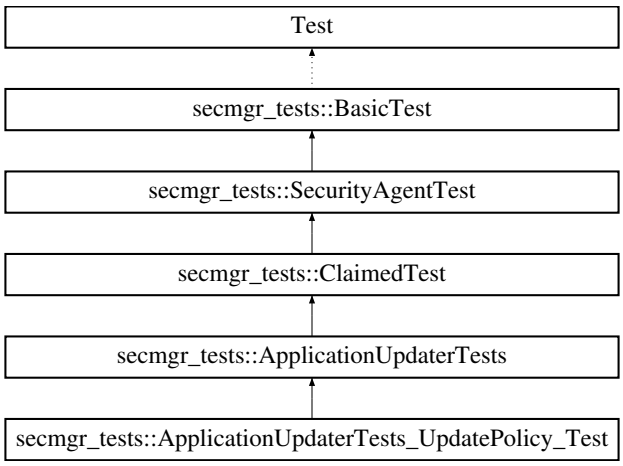
1. Claim and stop remote application.
2. Store a membership certificate for the application.
3. Update stored policy for the application.
4. Update stored identity certificate for the application.
5. Restart the remote application.
6. Wait for the updates to complete.
7. Ensure the membership certificate is installed correctly.
8. Ensure the policy is updated correctly.
9. Ensure the identity certificate is updated correctly.
10. Stop the remote application again.
11. Reset the remote application from storage.
12. Restart the remote application.
13. Check whether the remote application is CLAIMABLE again.

The documentation for this class was generated from the following file:

- ApplicationUpdaterTests.cc

5.16 secmgr_tests::ApplicationUpdaterTests_UpdatePolicy_Test Class Reference

Inheritance diagram for secmgr_tests::ApplicationUpdaterTests_UpdatePolicy_Test:



Additional Inherited Members

5.16.1 Detailed Description

Test Update a policy for an offline application and check whether it was successfully updated when it comes back online.

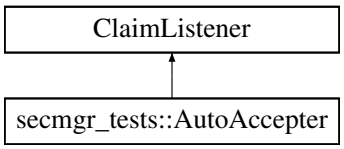
1. Claim and stop remote application.
2. Update the stored policy of the application.
3. Restart the remote application.
4. Wait for the updates to complete.
5. Ensure the policy is correctly installed.

The documentation for this class was generated from the following file:

- ApplicationUpdaterTests.cc

5.17 secmgr_tests::AutoAccepter Class Reference

Inheritance diagram for secmgr_tests::AutoAccepter:



Public Attributes

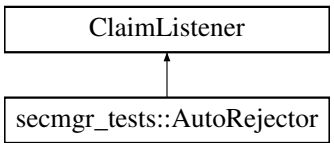
- Manifest **lastManifest**

The documentation for this class was generated from the following file:

- TestUtil.h

5.18 secmgr_tests::AutoRejector Class Reference

Inheritance diagram for secmgr_tests::AutoRejector:

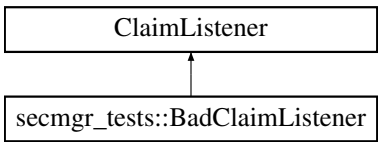


The documentation for this class was generated from the following file:

- ClaimingTests.cc

5.19 secmgr_tests::BadClaimListener Class Reference

Inheritance diagram for secmgr_tests::BadClaimListener:



Public Member Functions

- **BadClaimListener** (const GUID128 &guid)
- QStatus **ApproveManifestAndSelectSessionType** (ClaimContext &ctx)

Public Attributes

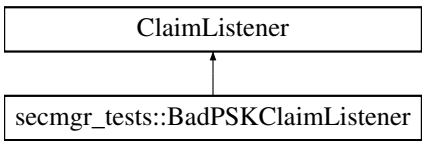
- bool **callSetClaimType**
- bool **callApproveManifest**
- bool **setPsk**
- QStatus **retVal**
- GUID128 **psk**

The documentation for this class was generated from the following file:

- ClaimingTests.cc

5.20 secmgr_tests::BadPSKClaimListener Class Reference

Inheritance diagram for secmgr_tests::BadPSKClaimListener:



Public Member Functions

- QStatus **ApproveManifestAndSelectSessionType** (ClaimContext &ctx)

Public Attributes

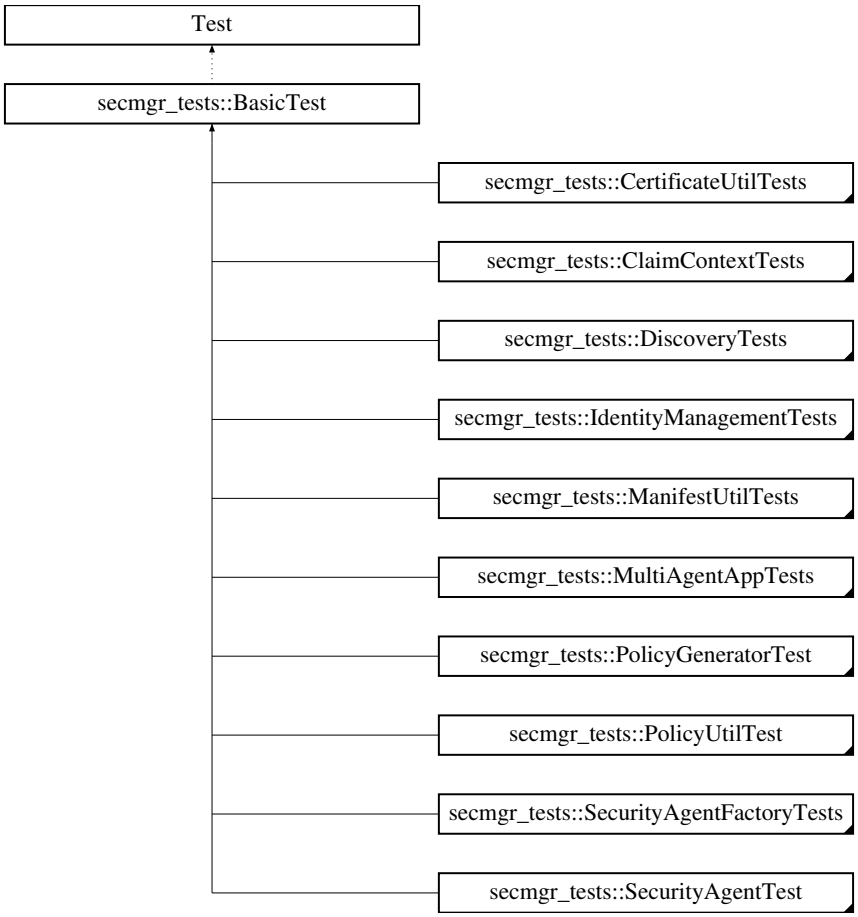
- GUID128 **localPsk**

The documentation for this class was generated from the following file:

- ClaimingTests.cc

5.21 secmgr_tests::BasicTest Class Reference

Inheritance diagram for secmgr_tests::BasicTest:



Public Member Functions

- bool **WaitForState** (PermissionConfigurator::ApplicationState newApplicationState, ApplicationSyncState updateState=SYNC_UNKNOWN)
- bool **CheckRemotePolicy** (PermissionPolicy &expectedPolicy)
- bool **CheckStoredPolicy** (PermissionPolicy &expectedPolicy)
- bool **CheckPolicy** (PermissionPolicy &expectedPolicy)
- bool **CheckDefaultPolicy** ()
- bool **CheckRemoteIdentity** (IdentityInfo &expectedIdentity, Manifest &expectedManifest, IdentityCertificate &remoteIdentity, Manifest &remoteManifest)
- bool **CheckIdentity** (IdentityInfo &expectedIdentity, Manifest &expectedManifest)
- bool **CheckMemberships** (vector< GroupInfo > expectedGroups)
- bool **CheckSyncState** (ApplicationSyncState updateState)
- bool **WaitForUpdatesCompleted** ()
- bool **WaitForSyncError** (SyncErrorType type, QStatus status)
- bool **WaitForEvents** (size_t numOfEvents)
- bool **CheckUnexpectedSyncErrors** ()
- bool **WaitForManifestUpdate** (ManifestUpdate &manifestUpdate)
- bool **CheckUnexpectedManifestUpdates** ()
- QStatus **Reset** (const OnlineApplication &app)
- QStatus **GetMembershipSummaries** (const OnlineApplication &app, vector< MembershipSummary > &summaries)

- QStatus **GetPolicyVersion** (const OnlineApplication &app, uint32_t &version)
- QStatus **GetIdentity** (const OnlineApplication &app, IdentityCertificateChain &idCertChain)
- QStatus **GetClaimCapabilities** (const OnlineApplication &app, PermissionConfigurator::ClaimCapabilities &claimCaps, PermissionConfigurator::ClaimCapabilityAdditionalInfo &claimCapInfo)
- virtual shared_ptr < AgentCAStorage > & **GetAgentCAStorage** ()
- void **InitSecAgent** ()
- void **RemoveSecAgent** ()

Public Attributes

- shared_ptr< SecurityAgent > **secMgr**
- BusAttachment * **ba**
- shared_ptr< UIStorage > **storage**
- shared_ptr< AgentCAStorage > **ca**
- OnlineApplication **lastAppInfo**
- AutoAccepter **aa**
- PolicyGenerator * **pg**
- ProxyObjectManager * **proxyObjectManager**
- Mutex **secAgentLock**

Protected Member Functions

- virtual void **SetUp** ()
- virtual void **TearDown** ()

Protected Attributes

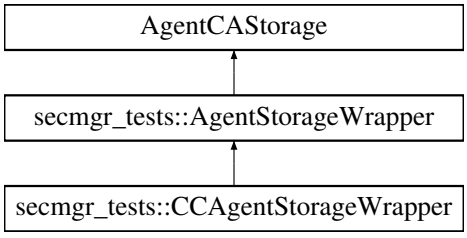
- Condition **sem**
- Mutex **lock**
- Condition **errorSem**
- Mutex **errorLock**
- Condition **manifestSem**
- Mutex **manifestLock**
- TestApplicationListener * **tal**
- TestAboutListener **testAboutListener**

The documentation for this class was generated from the following files:

- TestUtil.h
- TestUtil.cc

5.22 secmgr_tests::CCAgentStorageWrapper Class Reference

Inheritance diagram for secmgr_tests::CCAgentStorageWrapper:



Public Member Functions

- **CCAgentStorageWrapper** (shared_ptr< AgentCAStorage > &_ca, shared_ptr< UIStorage > &_storage)
- QStatus **UpdatesCompleted** (Application &app, uint64_t &updateID)
- void **SetAction** (Application _app, Action _action)
- void **SetAction** (Application _app, PermissionPolicy _policy)
- void **SetAction** (Application _app, GroupInfo _group)
- void **BlockNothingAction** ()
- void **UnblockNothingAction** ()

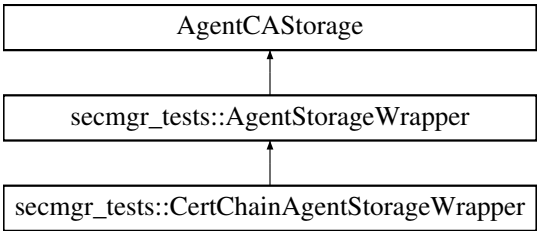
Additional Inherited Members

The documentation for this class was generated from the following file:

- ConcurrentUpdateTests.cc

5.23 secmgr_tests::CertChainAgentStorageWrapper Class Reference

Inheritance diagram for secmgr_tests::CertChainAgentStorageWrapper:



Public Member Functions

- **CertChainAgentStorageWrapper** (shared_ptr< AgentCAStorage > &_ca, CertificateX509 &rootCert)
- QStatus **RegisterAgent** (const KeyInfoNISTP256 &agentKey, const Manifest &manifest, GroupInfo &adminGroup, IdentityCertificateChain &identityCertificates, vector< MembershipCertificateChain > &adminGroupMemberships)
- virtual QStatus **GetMembershipCertificates** (const Application &app, vector< MembershipCertificateChain > &membershipCertificates) const
- QStatus **GetIdentityCertificatesAndManifest** (const Application &app, IdentityCertificateChain &identityCertificates, Manifest &manifest) const
- virtual QStatus **StartApplicationClaiming** (const Application &app, const IdentityInfo &idInfo, const Manifest &manifest, GroupInfo &adminGroup, IdentityCertificateChain &identityCertificates)

Public Attributes

- bool **addIdRootCert**
- bool **addMembershipRootCert**
- IdentityCertificateChain **agentIdChain**
- vector < MembershipCertificateChain > **agentMembershipCertificates**

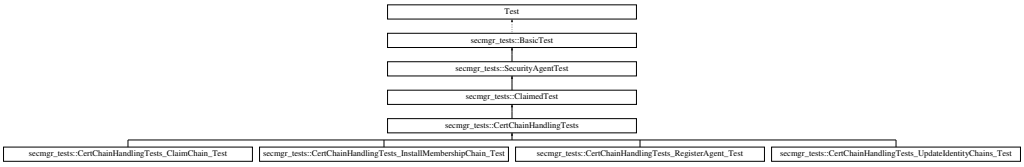
Additional Inherited Members

The documentation for this class was generated from the following file:

- CertChainHandlingTests.cc

5.24 secmgr_tests::CertChainHandlingTests Class Reference

Inheritance diagram for secmgr_tests::CertChainHandlingTests:



Public Member Functions

- shared_ptr< AgentCAStorage > & **GetAgentCAStorage** ()
- void **CheckIdentyCertificateChain** (IdentityCertificateChain chain, bool &failure, const char *function, int line)
- void **CheckMembershipSummaries** (bool &failure, const char *function, int line)

Public Attributes

- GroupInfo **groupInfo**
- vector< GUID128 > **policyGroups**
- CertificateX509 **rootCert**
- shared_ptr
  < CertChainAgentStorageWrapper > **wrappedCa**

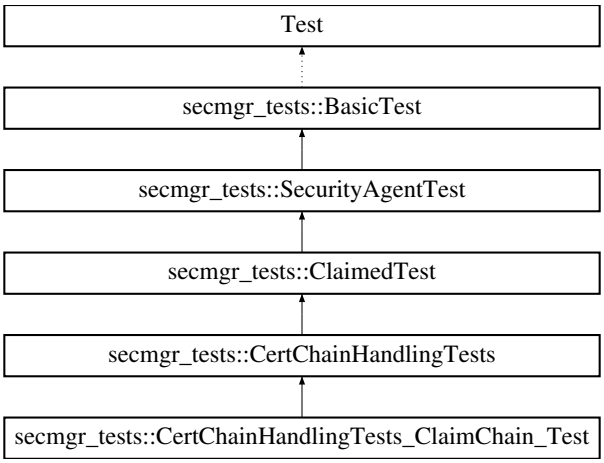
Additional Inherited Members

The documentation for this class was generated from the following file:

- CertChainHandlingTests.cc

5.25 secmgr_tests::CertChainHandlingTests_ClaimChain_Test Class Reference

Inheritance diagram for secmgr_tests::CertChainHandlingTests_ClaimChain_Test:



Additional Inherited Members

5.25.1 Detailed Description

Test Claim an application by presenting the agent an identity certificate chain.

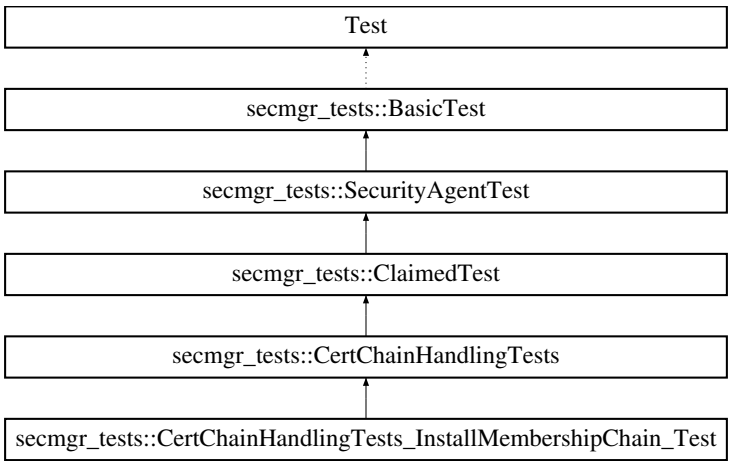
1. Claim an application and provide the agent an identity certificate chain
2. Check whether the application is CLAIMED.
3. Check if the application returns the full identity certificate chain

The documentation for this class was generated from the following file:

- CertChainHandlingTests.cc

5.26 secmgr_tests::CertChainHandlingTests_InstallMembershipChain_Test Class Reference

Inheritance diagram for secmgr_tests::CertChainHandlingTests_InstallMembershipChain_Test:



Additional Inherited Members

5.26.1 Detailed Description

Test Install the membership certificates by presenting the agent membership certificate chains.

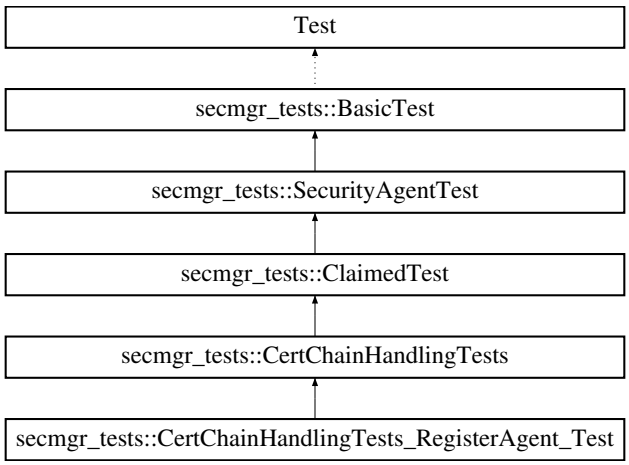
1. Install a membership certificate chain on application and provide the agent a membership certificate chain
2. Check whether the application is updated successfully.
3. Check if the application returns the full membership certificate chain
4. Install 2 more chains and verify they are found as well
5. Check if these chains can be removed and the application presents an empty list of membership certificates

The documentation for this class was generated from the following file:

- CertChainHandlingTests.cc

5.27 secmgr_tests::CertChainHandlingTests_RegisterAgent_Test Class Reference

Inheritance diagram for secmgr_tests::CertChainHandlingTests_RegisterAgent_Test:



Additional Inherited Members

5.27.1 Detailed Description

Test Validate that the register agent is able to handle a identity certificate chain and multiple membership certificate chains

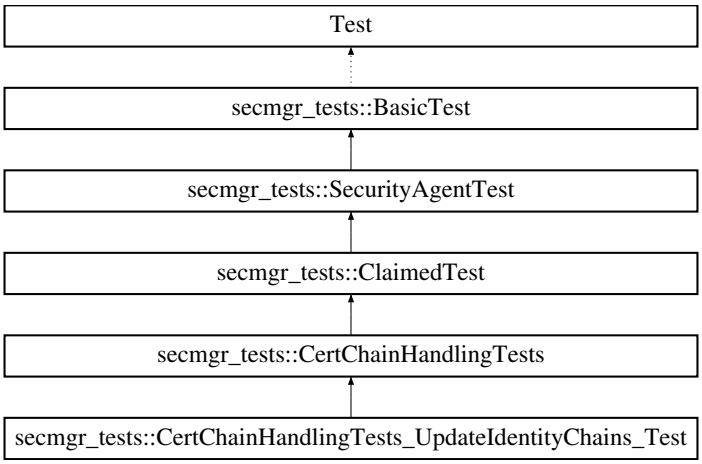
1. update the identity of an application and provide the agent an identity certificate chain
2. Check if the agent returns a correct membership list
3. Check if the agent returns the full identity certificate chain

The documentation for this class was generated from the following file:

- CertChainHandlingTests.cc

5.28 secmgr_tests::CertChainHandlingTests_UpdateIdentityChains_Test Class Reference

Inheritance diagram for secmgr_tests::CertChainHandlingTests_UpdateIdentityChains_Test:



Additional Inherited Members

5.28.1 Detailed Description

Test Update the identity of an already claimed application by presenting the agent an identity certificate chain.

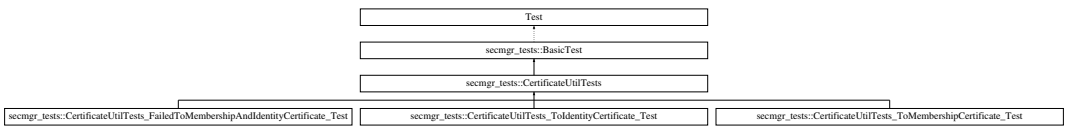
1. update the identity of an application and provide the agent an identity certificate chain
2. Check whether the application is updated successfully.
3. Check if the application returns the full identity certificate chain

The documentation for this class was generated from the following file:

- CertChainHandlingTests.cc

5.29 secmgr_tests::CertificateUtilTests Class Reference

Inheritance diagram for secmgr_tests::CertificateUtilTests:



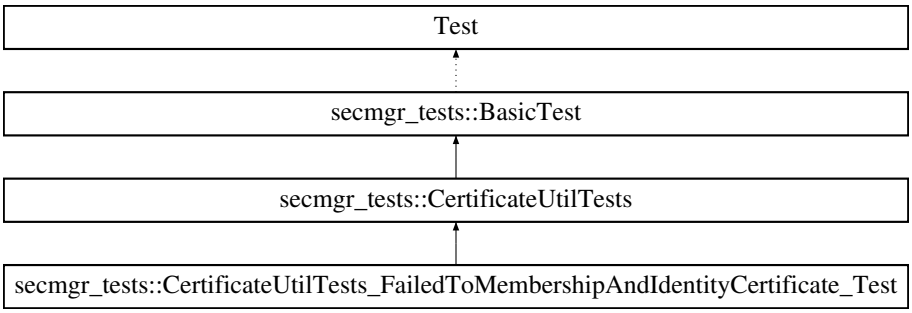
Additional Inherited Members

The documentation for this class was generated from the following file:

- CertificateUtilTests.cc

5.30 secmgr_tests::CertificateUtilTests_FailedToMembershipAndIdentityCertificate_Test Class Reference

Inheritance diagram for secmgr_tests::CertificateUtilTests_FailedToMembershipAndIdentityCertificate_Test:



Additional Inherited Members

5.30.1 Detailed Description

Test Verify the creation on membership and identity certificates fails if wrong validity is provided.

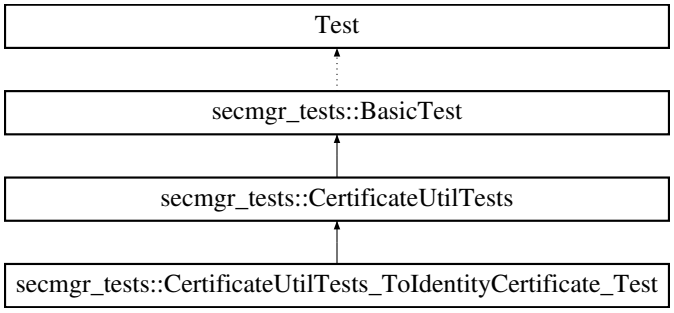
1. Create an Application.
2. Create a GroupInfo groupInfo and IdentityInfo identityInfo.
3. Use ToMembershipCertificate with zero validity period and make sure it fails (!= ER_OK).
4. Use ToIdentityCertificate zero validity period and make sure it fails (!= ER_OK).

The documentation for this class was generated from the following file:

- CertificateUtilTests.cc

5.31 secmgr_tests::CertificateUtilTests_ToIdentityCertificate_Test Class Reference

Inheritance diagram for secmgr_tests::CertificateUtilTests_ToIdentityCertificate_Test:



Additional Inherited Members

5.31.1 Detailed Description

Test Verify the creation on identity certificate.

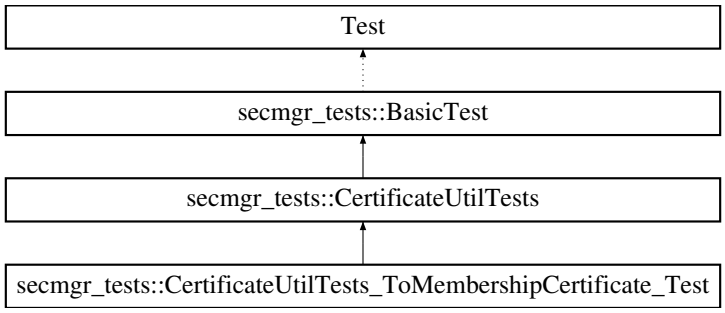
1. Create an Application with a valid full KeyInfoNISTP256 keyinfo.
2. Create a valid IdentityInfo identityInfo with a valid guid.
3. Declare a validityPeriod with a valid value.
4. Use ToIdentityCertificate and make sure it succeeds and returns an identityCert.
5. Verify that the identityCert fields are matching the ones passed on during creation.
6. Verify that identityCert is not a CA.
7. Verify that identityCert has no digest.
8. Verify that identityCert has a nullptr encoding.

The documentation for this class was generated from the following file:

- CertificateUtilTests.cc

5.32 secmgr_tests::CertificateUtilTests_ToMembershipCertificate_Test Class Reference

Inheritance diagram for secmgr_tests::CertificateUtilTests_ToMembershipCertificate_Test:



Additional Inherited Members

5.32.1 Detailed Description

Test Verify the creation on membership certificate.

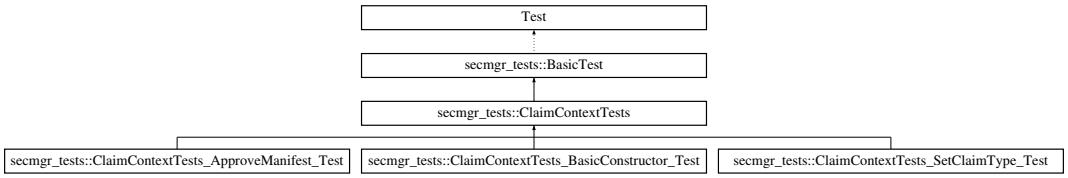
1. Create an Application with a valid full KeyInfoNISTP256 keyinfo.
2. Create a valid GroupInfo groupInfo with a valid guid.
3. Declare a validityPeriod with a valid value.
4. Use ToMembershipCertificate and make sure it succeeds and returns a membershipCert.
5. Verify that the membershipCert fields are matching the ones passed on during creation and that it is not a CA.
6. Verify that membershipCert has a nullptr encoding.

The documentation for this class was generated from the following file:

- CertificateUtilTests.cc

5.33 secmgr_tests::ClaimContextTests Class Reference

Inheritance diagram for secmgr_tests::ClaimContextTests:



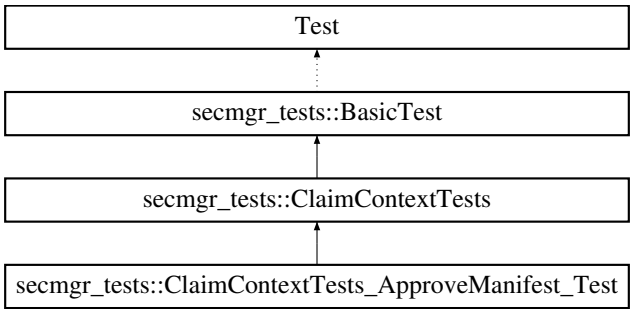
Additional Inherited Members

The documentation for this class was generated from the following file:

- ClaimContextTests.cc

5.34 secmgr_tests::ClaimContextTests_ApproveManifest_Test Class Reference

Inheritance diagram for secmgr_tests::ClaimContextTests_ApproveManifest_Test:



Additional Inherited Members

5.34.1 Detailed Description

Test Verify the ApproveManifest function is working as expected.

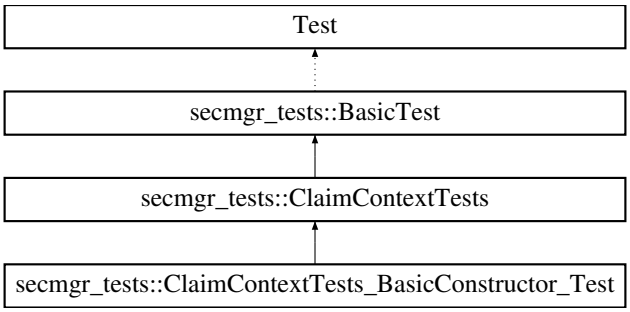
1. Create an ClaimContext object
2. Call the ApproveManifest function with different values and validate the result

The documentation for this class was generated from the following file:

- ClaimContextTests.cc

5.35 secmgr_tests::ClaimContextTests_BasicConstructor_Test Class Reference

Inheritance diagram for secmgr_tests::ClaimContextTests_BasicConstructor_Test:



Additional Inherited Members

5.35.1 Detailed Description

Test Verify the construction of a ClaimContext and its getters.

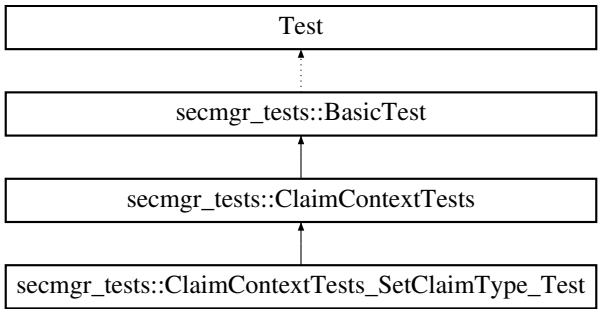
1. Create an ClaimContext object
2. Call the Get functions and check the return values

The documentation for this class was generated from the following file:

- ClaimContextTests.cc

5.36 secmgr_tests::ClaimContextTests_SetClaimType_Test Class Reference

Inheritance diagram for secmgr_tests::ClaimContextTests_SetClaimType_Test:



Additional Inherited Members

5.36.1 Detailed Description

Test Verify the SetClaimType function is working as expected.

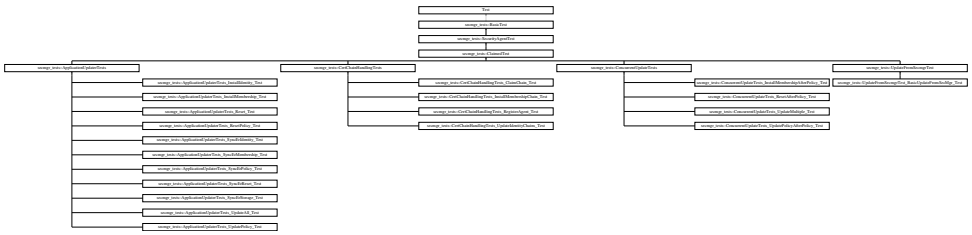
1. Create an ClaimContext object
2. Call the SetClaimType function with different values and validate the results

The documentation for this class was generated from the following file:

- ClaimContextTests.cc

5.37 secmgr_tests::ClaimedTest Class Reference

Inheritance diagram for secmgr_tests::ClaimedTest:



Public Member Functions

- void **SetUp** ()

Public Attributes

- IdentityInfo **idInfo**
- TestApplication **testApp**

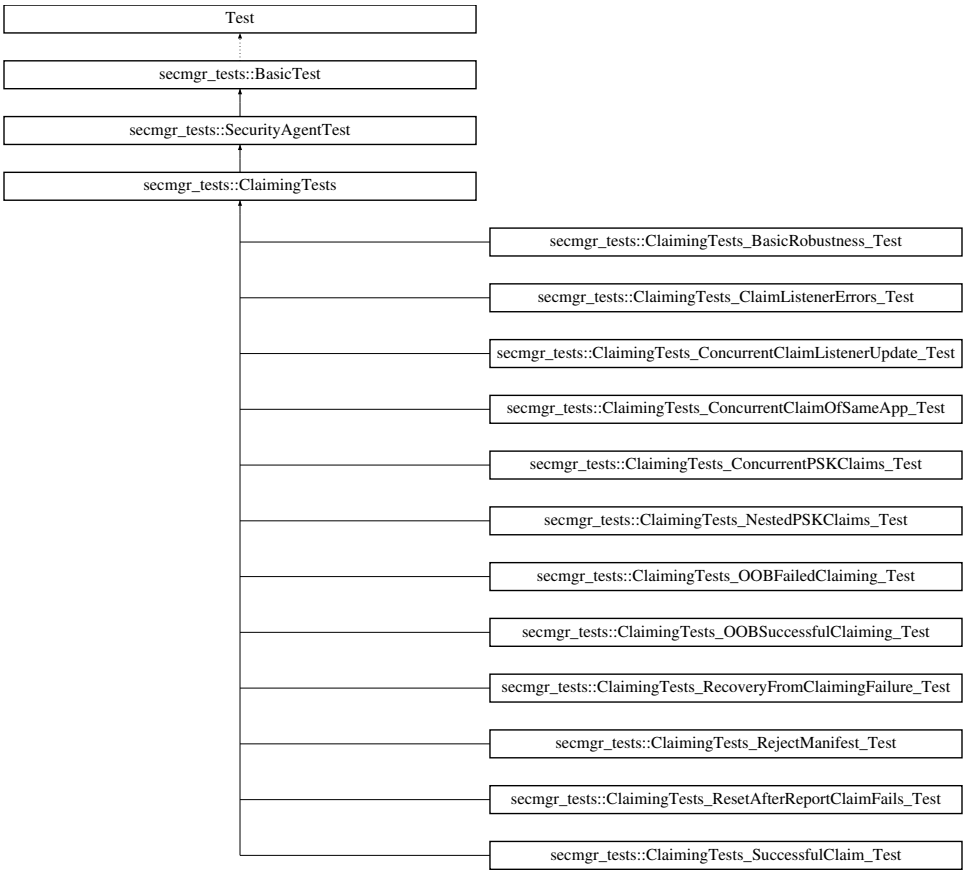
Additional Inherited Members

The documentation for this class was generated from the following file:

- TestUtil.h

5.38 secmgr_tests::ClaimingTests Class Reference

Inheritance diagram for secmgr_tests::ClaimingTests:



Public Member Functions

- shared_ptr< AgentCAStorage > & **GetAgentCAStorage** ()

Public Attributes

- shared_ptr< FailingStorageWrapper > **wrappedCA**

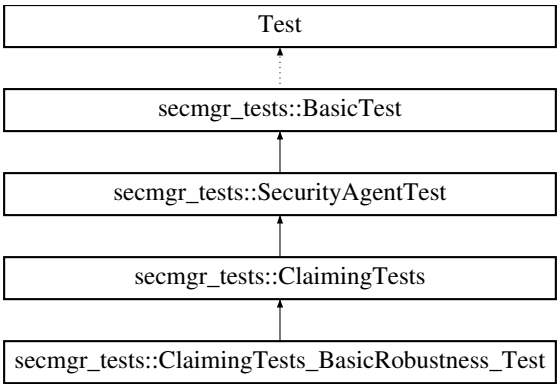
Additional Inherited Members

The documentation for this class was generated from the following file:

- ClaimingTests.cc

5.39 secmgr_tests::ClaimingTests_BasicRobustness_Test Class Reference

Inheritance diagram for secmgr_tests::ClaimingTests_BasicRobustness_Test:



Additional Inherited Members

5.39.1 Detailed Description

Test Basic robustness tests for the claiming, including input validation, unavailability of manifest listener/CA.

1. Claiming an off-line/unknown application should fail.
2. Claiming using an unknown identity should fail.
3. Claiming an application that is NOT_CLAIMABALE should fail.
4. Claiming an application that is CLAIMED should fail.
5. Claiming an application that NEED_UPDATE should fail.

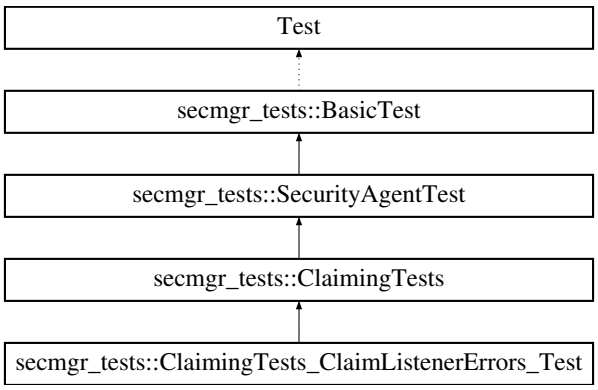
```
-# Claiming when no ClaimListener is set should fail.
```

The documentation for this class was generated from the following file:

- ClaimingTests.cc

5.40 secmgr_tests::ClaimingTests_ClaimListenerErrors_Test Class Reference

Inheritance diagram for secmgr_tests::ClaimingTests_ClaimListenerErrors_Test:



Additional Inherited Members

5.40.1 Detailed Description

Test Verify when the ClaimListener returns errors, these are handled correctly.

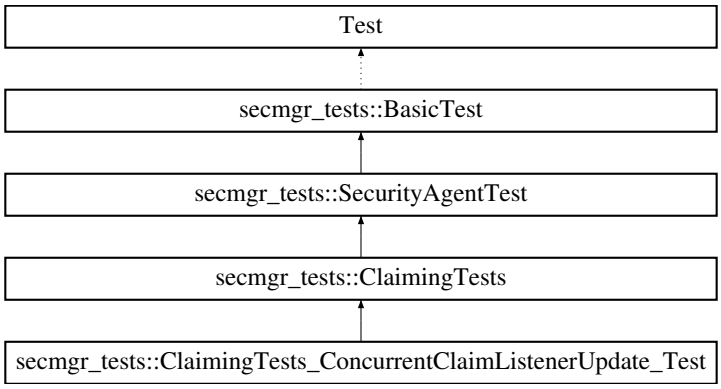
1. Start an application and make sure it's in the CLAIMABLE state
2. Try to claim it and trigger error conditions in the ClaimListener.
3. Verify that the application remains CLAIMABLE.

The documentation for this class was generated from the following file:

- ClaimingTests.cc

5.41 secmgr_tests::ClaimingTests_ConcurrentClaimListenerUpdate_Test Class Reference

Inheritance diagram for secmgr_tests::ClaimingTests_ConcurrentClaimListenerUpdate_Test:



Additional Inherited Members

5.41.1 Detailed Description

Test Changing the manifest listener when being in the callback of the original manifest listener should work.

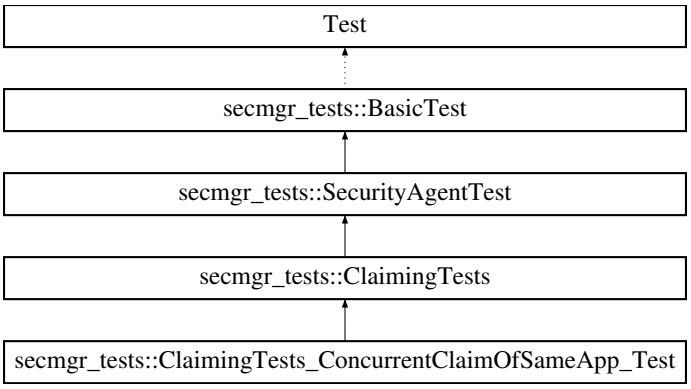
1. Claim a CLAIMABLE application with a known identity.
2. While the claim listener is called to approve the manifest, a new manifest listener is installed to reject the manifest.
3. The original listener accepts the manifest.
4. The application should be claimed.
5. Start a new application and try claiming it.
6. The manifest should be rejected and the claiming should fail.
7. Make sure the new application is still claimable.

The documentation for this class was generated from the following file:

- ClaimingTests.cc

5.42 secmgr_tests::ClaimingTests_ConcurrentClaimOfSameApp_Test Class Reference

Inheritance diagram for secmgr_tests::ClaimingTests_ConcurrentClaimOfSameApp_Test:



Additional Inherited Members

5.42.1 Detailed Description

Test Verify that the agent rejects claim of an application when it is already claiming that application

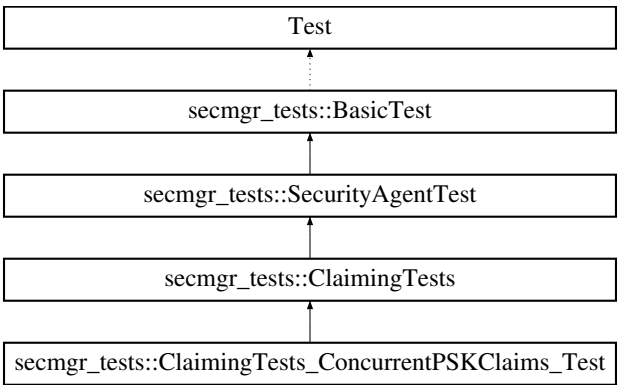
1. Start an application and make sure it is claimable.
2. Claim it, but make sure to claim it again while the first claim is ongoing
3. Verify that the application becomes claimed and the second claim fails

The documentation for this class was generated from the following file:

- ClaimingTests.cc

5.43 secmgr_tests::ClaimingTests_ConcurrentPSKClaims_Test Class Reference

Inheritance diagram for secmgr_tests::ClaimingTests_ConcurrentPSKClaims_Test:



Additional Inherited Members

5.43.1 Detailed Description

Test Verify that the agent concurrently can claim multiple applications.

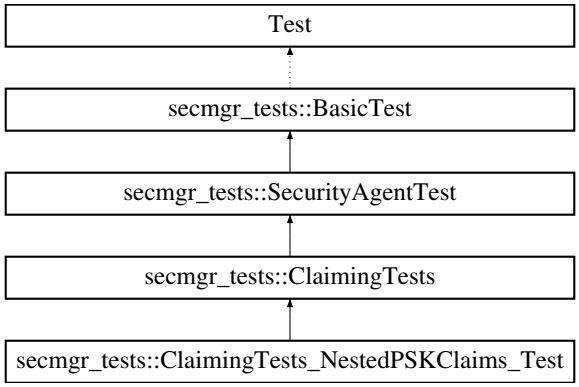
1. Start multiple applications and try to claim them in parallel
2. Verify that all applications become claimed.

The documentation for this class was generated from the following file:

- ClaimingTests.cc

5.44 secmgr_tests::ClaimingTests_NestedPSKClaims_Test Class Reference

Inheritance diagram for secmgr_tests::ClaimingTests_NestedPSKClaims_Test:



Additional Inherited Members

5.44.1 Detailed Description

Test Verify when the ClaimListener claim another application, these extra claims are handled correctly.

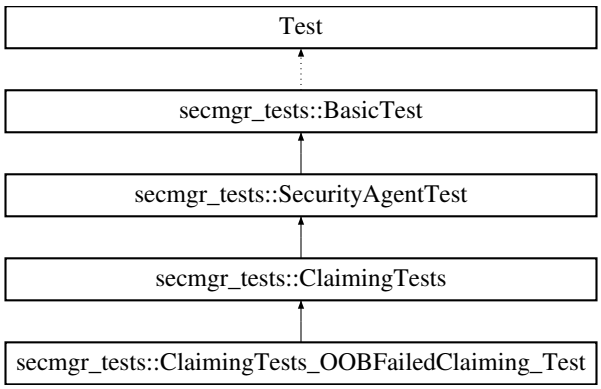
1. Start multiple application and make sure it's in the CLAIMABLE state
2. Try to claim them nesting the claim calls
3. Verify that the applications are CLAIMED.

The documentation for this class was generated from the following file:

- ClaimingTests.cc

5.45 secmgr_tests::ClaimingTests_OOBFailedClaiming_Test Class Reference

Inheritance diagram for secmgr_tests::ClaimingTests_OOBFailedClaiming_Test:



Additional Inherited Members

5.45.1 Detailed Description

Test Verify claiming with Out-Of-Band (OOB) fails when wrong PSK is used.

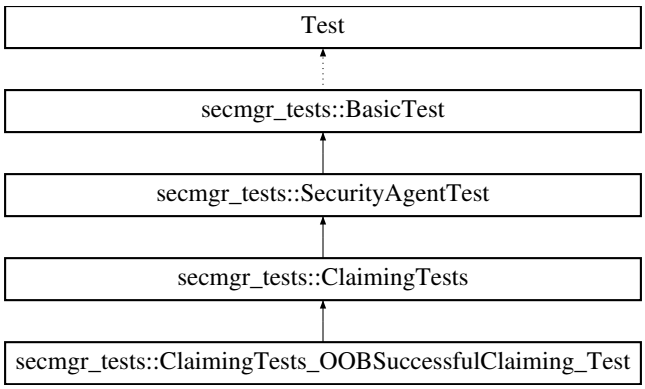
1. Start an application and make sure it's in the CLAIMABLE state with PSK preference; i.e., OOB.
2. Make sure that the security agent has generated the PSK.
3. Verify the the application uses a different PSK for OOB claiming.
4. Verify that the application is still CLAIMABLE and online and that claiming has failed.
5. Repeat the scenario where the PSK is generated by the application instead of the security agent and make sure PSK claiming fails.

The documentation for this class was generated from the following file:

- ClaimingTests.cc

5.46 secmgr_tests::ClaimingTests_OOBSuccessfulClaiming_Test Class Reference

Inheritance diagram for secmgr_tests::ClaimingTests_OOBSuccessfulClaiming_Test:



Additional Inherited Members

5.46.1 Detailed Description

Test Verify claiming with Out-Of-Band (OOB) succeeds.

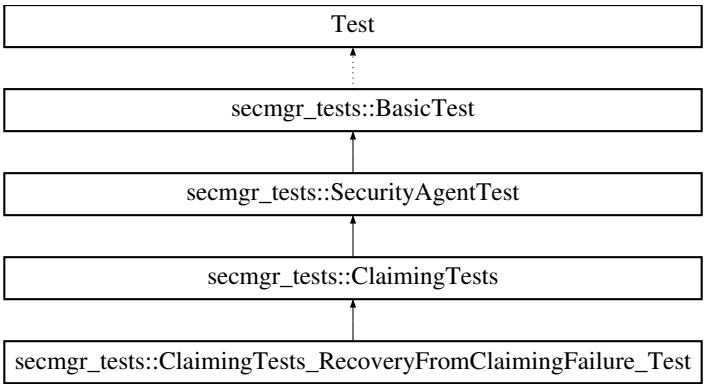
1. Start an application and make sure it's in the CLAIMABLE state with PSK preference; i.e., OOB.
2. Make sure that the application has generated the PSK.
3. Verify the the security agent uses the same PSK for OOB claiming and accepts the manifest.
4. Verify that the application is CLAIMED and online.
5. Reset/remove the application and make sure it's claimable again and repeat the scenario.
6. Verify that claiming was successful and that the application is in CLAIMED state and online.

The documentation for this class was generated from the following file:

- ClaimingTests.cc

5.47 secmgr_tests::ClaimingTests_RecoveryFromClaimingFailure_Test Class Reference

Inheritance diagram for secmgr_tests::ClaimingTests_RecoveryFromClaimingFailure_Test:



Additional Inherited Members

5.47.1 Detailed Description

Test Recovery from failure of notifying the CA of failure of claiming an application should be graceful.

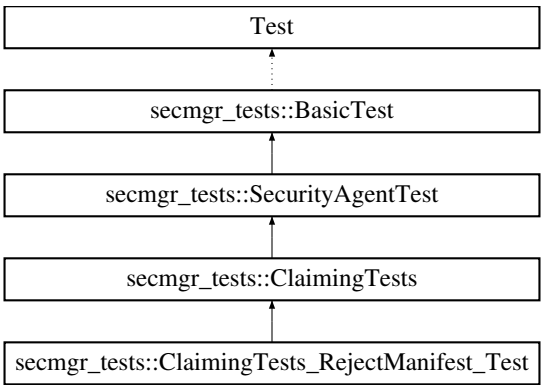
1. Start a test application.
2. Install a manifest listener that stops the application before accepting the manifest, which will make the claiming fail.
3. Make sure the UpdatesCompleted to storage fails.
4. Claim the application and check that this fails.
5. Wait for a sync error for the application.

The documentation for this class was generated from the following file:

- ClaimingTests.cc

5.48 secmgr_tests::ClaimingTests_RejectManifest_Test Class Reference

Inheritance diagram for secmgr_tests::ClaimingTests_RejectManifest_Test:



Additional Inherited Members

5.48.1 Detailed Description

Test Reject the manifest during claiming and check whether the application becomes CLAIMABLE again.

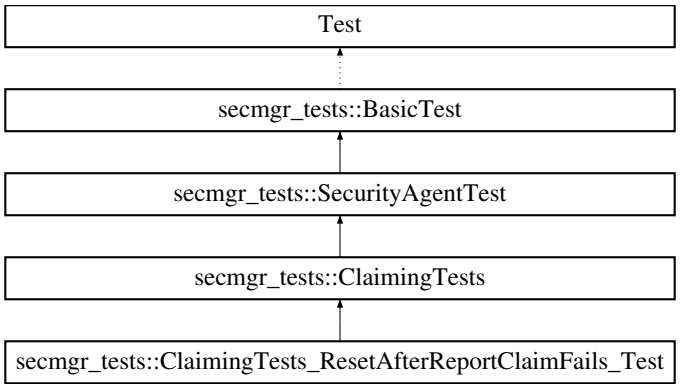
1. Claim the remote application.
2. Reject the manifest.
3. Check whether the agent returns an ER_MANIFEST_REJECTED error.
4. Check whether the application remains CLAIMABLE.

The documentation for this class was generated from the following file:

- ClaimingTests.cc

5.49 secmgr_tests::ClaimingTests_ResetAfterReportClaimFails_Test Class Reference

Inheritance diagram for secmgr_tests::ClaimingTests_ResetAfterReportClaimFails_Test:



Additional Inherited Members

5.49.1 Detailed Description

Test Verify that the agent resets the application after it claims it, but receives an error from the storage

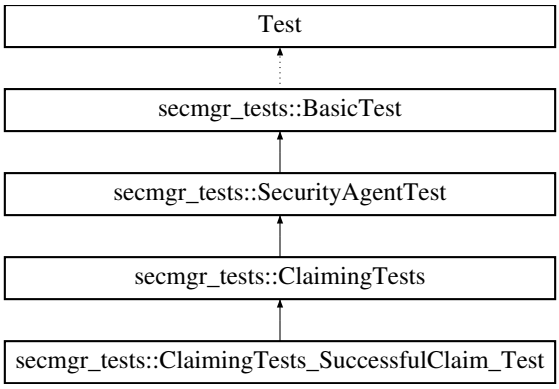
1. Start an application and make sure it is claimable.
2. Claim it, but make sure that the storage FinishApplicationClaiming fails
3. Verify that the application becomes claimed for a short while and then claimable again

The documentation for this class was generated from the following file:

- ClaimingTests.cc

5.50 secmgr_tests::ClaimingTests_SuccessfulClaim_Test Class Reference

Inheritance diagram for secmgr_tests::ClaimingTests_SuccessfulClaim_Test:



Additional Inherited Members

5.50.1 Detailed Description

Test Claim an application and check that it becomes CLAIMED.

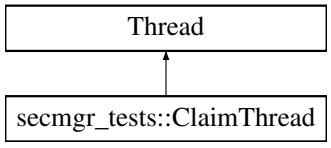
1. Start the application.
2. Make sure the application is in a CLAIMABLE state.
3. Claim the application.
4. Accept the manifest of the application.
5. Check whether the application becomes CLAIMED.

The documentation for this class was generated from the following file:

- ClaimingTests.cc

5.51 secmgr_tests::ClaimThread Class Reference

Inheritance diagram for secmgr_tests::ClaimThread:



Public Member Functions

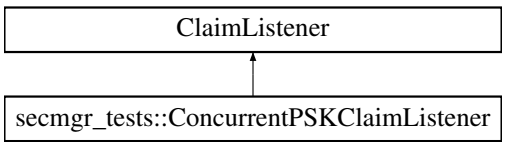
- **ClaimThread** (const IdentityInfo &_idInfo, const OnlineApplication &_app, const shared_ptr< SecurityAgent > &_secMgr)
- virtual ThreadReturn STDCALL **Run** (void *arg)

The documentation for this class was generated from the following file:

- ClaimingTests.cc

5.52 secmgr_tests::ConcurrentPSKClaimListener Class Reference

Inheritance diagram for secmgr_tests::ConcurrentPSKClaimListener:



Public Member Functions

- **ConcurrentPSKClaimListener** (vector< shared_ptr< TestApplication > > &_testapps)
- QStatus **ApproveManifestAndSelectSessionType** (ClaimContext &ctx)

Public Attributes

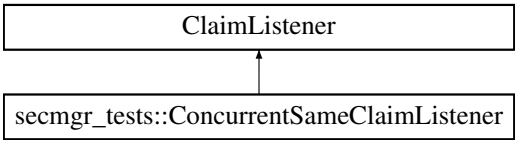
- vector< shared_ptr < TestApplication > > **testapps**

The documentation for this class was generated from the following file:

- ClaimingTests.cc

5.53 secmgr_tests::ConcurrentSameClaimListener Class Reference

Inheritance diagram for secmgr_tests::ConcurrentSameClaimListener:



Public Member Functions

- **ConcurrentSameClaimListener** (const IdentityInfo &_idInfo, const shared_ptr< SecurityAgent > &_secMgr)
- QStatus **ApproveManifestAndSelectSessionType** (ClaimContext &ctx)

Public Attributes

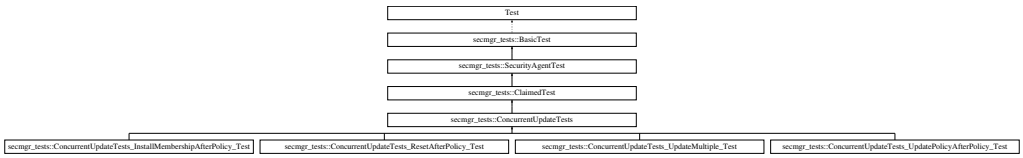
- bool **checked**
- IdentityInfo **idInfo**
- shared_ptr< SecurityAgent > **secMgr**

The documentation for this class was generated from the following file:

- ClaimingTests.cc

5.54 secmgr_tests::ConcurrentUpdateTests Class Reference

Inheritance diagram for secmgr_tests::ConcurrentUpdateTests:



Public Member Functions

- void **TearDown** ()
- shared_ptr< AgentCAStorage > & **GetAgentCAStorage** ()

Public Attributes

- GroupInfo **groupInfo**
- PermissionPolicy **policy**
- vector< GUID128 > **policyGroups**
- shared_ptr< CCAgentStorageWrapper > **wrappedCa**

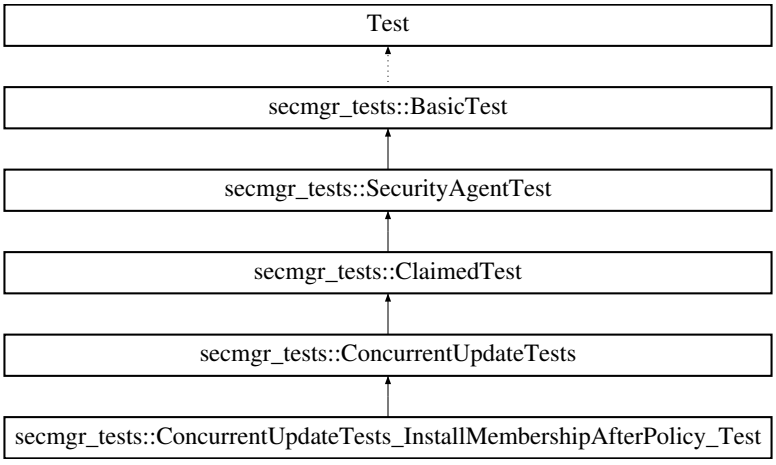
Additional Inherited Members

The documentation for this class was generated from the following file:

- ConcurrentUpdateTests.cc

5.55 secmgr_tests::ConcurrentUpdateTests_InstallMembershipAfterPolicy_Test Class Reference

Inheritance diagram for secmgr_tests::ConcurrentUpdateTests_InstallMembershipAfterPolicy_Test:



Additional Inherited Members

5.55.1 Detailed Description

Test Install a membership certificate on an application while updating its policy and check whether both are installed correctly.

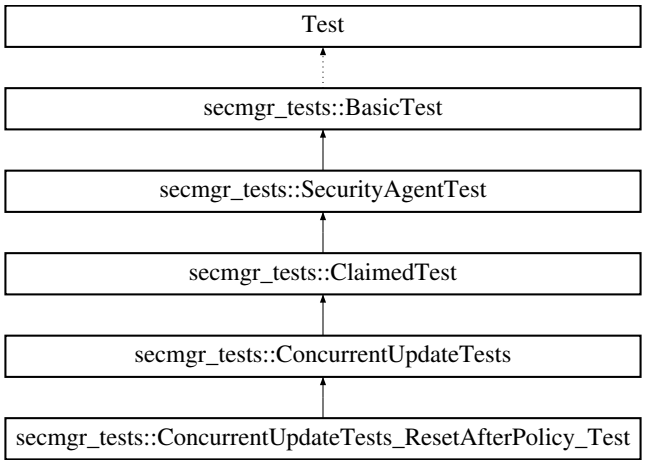
1. Claim an application.
2. Update the policy of the application.
3. While updating the policy, store a membership certificate for the policy.
4. Wait for the updates to complete.
5. Check whether the policy was updated correctly.
6. Check whether the membership was installed correctly.

The documentation for this class was generated from the following file:

- ConcurrentUpdateTests.cc

5.56 secmgr_tests::ConcurrentUpdateTests_ResetAfterPolicy_Test Class Reference

Inheritance diagram for secmgr_tests::ConcurrentUpdateTests_ResetAfterPolicy_Test:



Additional Inherited Members

5.56.1 Detailed Description

Test Reset an application while updating its policy and check whether it is CLAIMABLE.

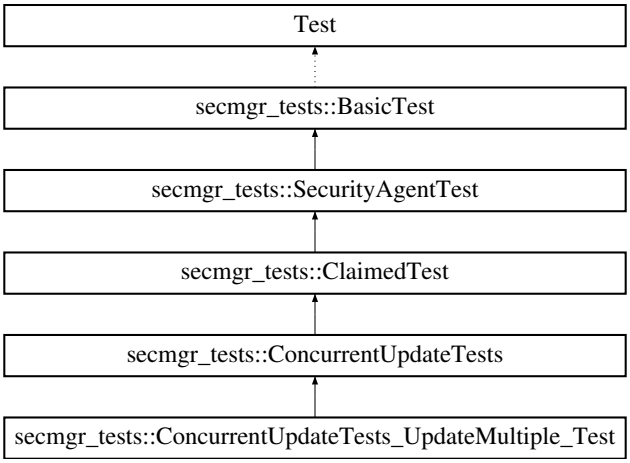
1. Claim an application.
2. Update the policy of the application.
3. While updating the policy, reset the application using the security agent.
4. Check whether the application is CLAIMABLE.

The documentation for this class was generated from the following file:

- ConcurrentUpdateTests.cc

5.57 secmgr_tests::ConcurrentUpdateTests_UpdateMultiple_Test Class Reference

Inheritance diagram for secmgr_tests::ConcurrentUpdateTests_UpdateMultiple_Test:



Additional Inherited Members

5.57.1 Detailed Description

Test Install a membership certificate on an application and update its policy while updating its policy, and check whether the last policy and the membership certificate have been installed successfully.

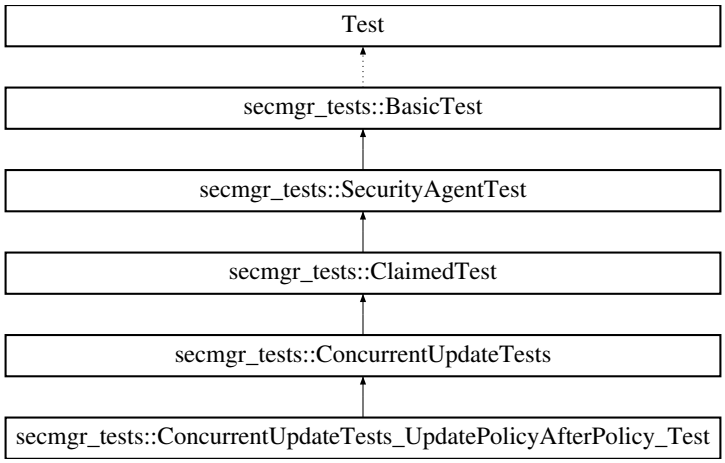
1. Claim an application.
2. Update the policy of the application.
3. While updating the policy, install a membership certificate.
4. While installing the membership certificate, update its policy.
5. Wait for the updates to complete.
6. Check whether the last policy is installed correctly.
7. Check whether the membership certificate was installed correctly.

The documentation for this class was generated from the following file:

- ConcurrentUpdateTests.cc

5.58 secmgr_tests::ConcurrentUpdateTests_UpdatePolicyAfterPolicy_Test Class Reference

Inheritance diagram for secmgr_tests::ConcurrentUpdateTests_UpdatePolicyAfterPolicy_Test:



Additional Inherited Members

5.58.1 Detailed Description

Test Update the policy of an application while updating its policy and check whether the last policy is installed successfully.

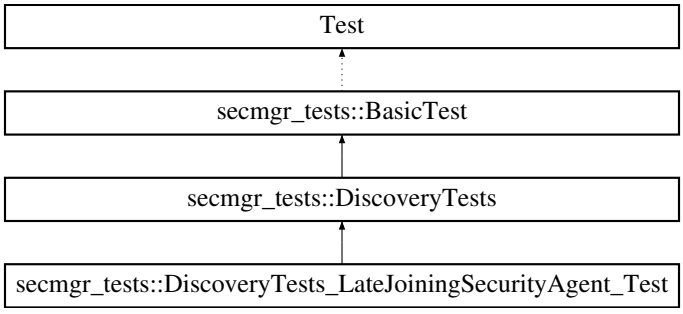
1. Claim an application.
2. Update the policy of the application.
3. While updating the policy, store another policy for the application.
4. Wait for the updates to complete.
5. Check whether the last policy is installed correctly.

The documentation for this class was generated from the following file:

- ConcurrentUpdateTests.cc

5.59 secmgr_tests::DiscoveryTests Class Reference

Inheritance diagram for secmgr_tests::DiscoveryTests:



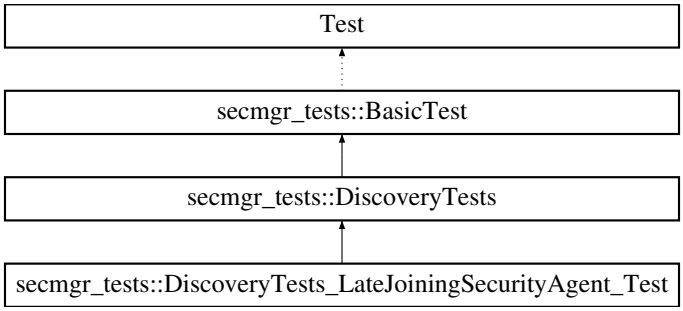
Additional Inherited Members

The documentation for this class was generated from the following file:

- DiscoveryTests.cc

5.60 secmgr_tests::DiscoveryTests_LateJoiningSecurityAgent_Test Class Reference

Inheritance diagram for secmgr_tests::DiscoveryTests_LateJoiningSecurityAgent_Test:



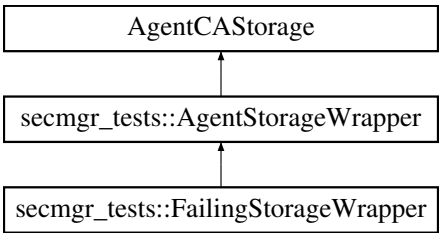
Additional Inherited Members

The documentation for this class was generated from the following file:

- DiscoveryTests.cc

5.61 secmgr_tests::FailingStorageWrapper Class Reference

Inheritance diagram for secmgr_tests::FailingStorageWrapper:



Public Member Functions

- **FailingStorageWrapper** (shared_ptr< AgentCAStorage > &_ca, shared_ptr< UIStorage > &_storage)
- QStatus **FinishApplicationClaiming** (const Application &app, QStatus status)
- QStatus **UpdatesCompleted** (Application &app, uint64_t &updateID)

Public Attributes

- bool **failOnUpdatesCompleted**
- bool **failOnFinishApplicationClaiming**

Additional Inherited Members

The documentation for this class was generated from the following file:

- AgentStorageWrapper.h

5.62 secmgr_tests::GroupManagementTests Class Reference

Inheritance diagram for secmgr_tests::GroupManagementTests:

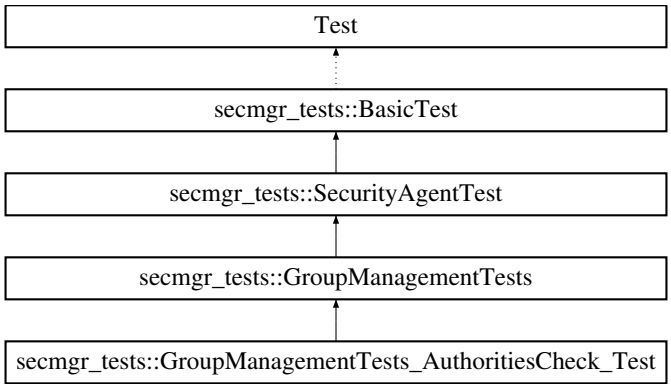
Additional Inherited Members

The documentation for this class was generated from the following file:

- GroupManagementTests.cc

5.63 secmgr_tests::GroupManagementTests_AuthoritiesCheck_Test Class Reference

Inheritance diagram for secmgr_tests::GroupManagementTests_AuthoritiesCheck_Test:



Additional Inherited Members

5.63.1 Detailed Description

Test Check whether more than one group authority can be supported.

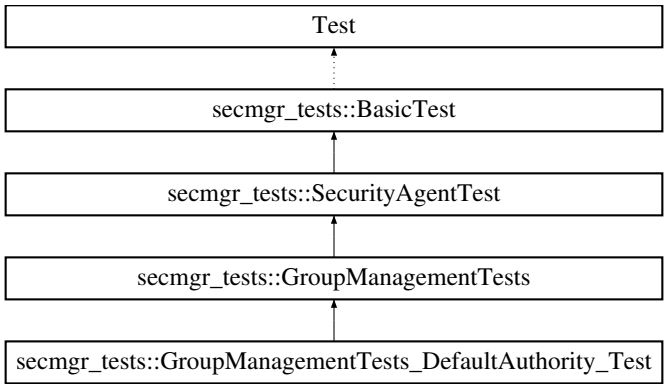
1. Create a GroupInfo (group0) object with an empty authority
2. Store it successfully and make sure it gets an authority set.
3. Create another GroupInfo (group1) and assign it group0's GUID but a different authority.
4. Verify storing of group1 fails as it uses group0's GUID.
5. Change group1 to use the same authority as in group0.
6. Verify updating storage with group1 succeeds and that both groups are identical.
7. Create a different group2 with a different GUID and Authority.
8. Make sure its storage and retrieval succeed and that it is different than both previous groups.

The documentation for this class was generated from the following file:

- GroupManagementTests.cc

5.64 secmgr_tests::GroupManagementTests_DefaultAuthority_Test Class Reference

Inheritance diagram for secmgr_tests::GroupManagementTests_DefaultAuthority_Test:



Additional Inherited Members

5.64.1 Detailed Description

Test Check whether the default group authority is added on all Group methods.

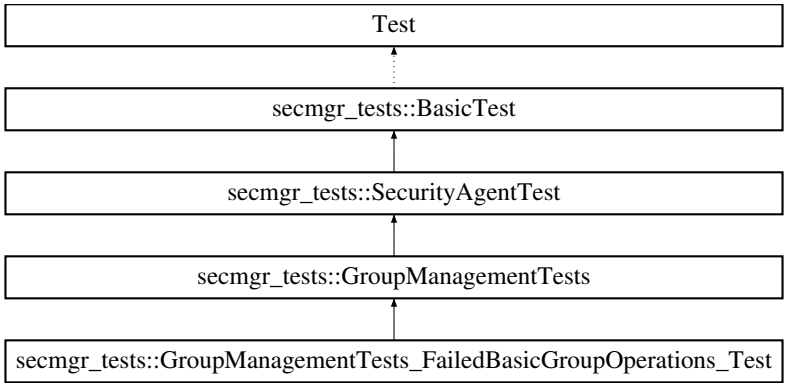
1. Create a GroupInfo object.
2. Store the GroupInfo object and verify the authority is set.
3. Create another GroupInfo object and fill in only the guid.
4. Check if the original GroupInfo object can be retrieved.
5. Create another GroupInfo object and fill in only the guid.
6. Check if the original GroupInfo object can be removed.

The documentation for this class was generated from the following file:

- GroupManagementTests.cc

5.65 secmgr_tests::GroupManagementTests_FailedBasicGroupOperations_Test Class Reference

Inheritance diagram for secmgr_tests::GroupManagementTests_FailedBasicGroupOperations_Test:



Additional Inherited Members

5.65.1 Detailed Description

Test Retrieval and deletion of unknown groups should fail.

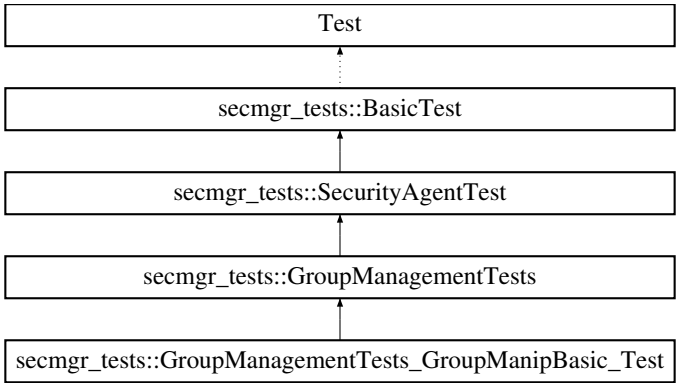
1. Try to get an unknown group and make sure this fails.
2. Try to remove an unknown group and make sure this fails.
3. Try to get all managed groups and make sure the vector is empty.

The documentation for this class was generated from the following file:

- GroupManagementTests.cc

5.66 secmgr_tests::GroupManagementTests_GroupManipBasic_Test Class Reference

Inheritance diagram for secmgr_tests::GroupManagementTests_GroupManipBasic_Test:



Additional Inherited Members

5.66.1 Detailed Description

Test Store, retrieve and delete a group from storage.

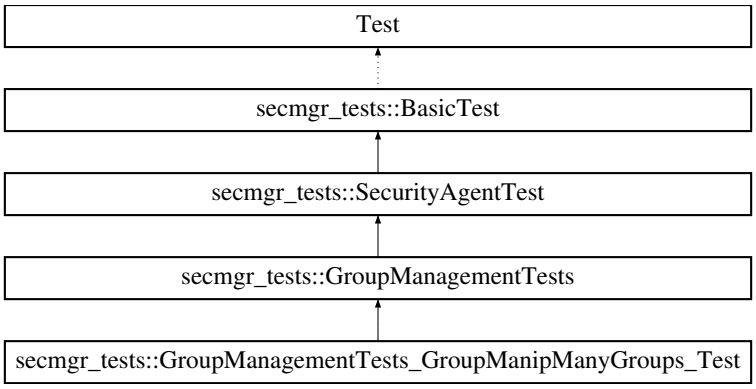
1. Define a valid security group and store it.
2. Retrieve the group from storage and check whether it matches the stored group.
3. Remove the security group.
4. Retrieving the security group should fail.

The documentation for this class was generated from the following file:

- GroupManagementTests.cc

5.67 secmgr_tests::GroupManagementTests_GroupManipManyGroups_Test Class Reference

Inheritance diagram for secmgr_tests::GroupManagementTests_GroupManipManyGroups_Test:



Additional Inherited Members

5.67.1 Detailed Description

Test Store, retrieve and delete many security groups from storage.

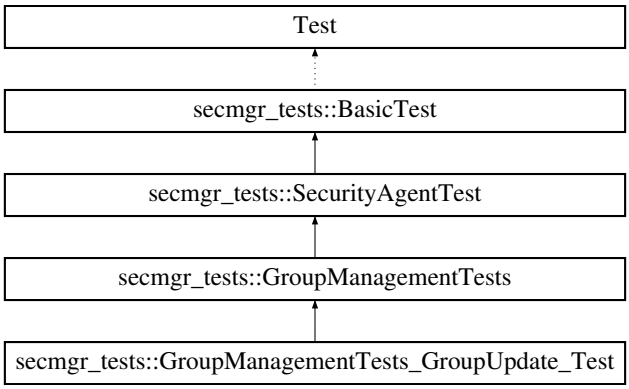
1. Define a series of security groups and store them one by one.
2. Retrieve all groups from storage and count whether all have been stored correctly.
3. Remove all groups one by one from storage.
4. Retrieve all groups from storage and make sure none are returned.

The documentation for this class was generated from the following file:

- GroupManagementTests.cc

5.68 secmgr_tests::GroupManagementTests_GroupUpdate_Test Class Reference

Inheritance diagram for secmgr_tests::GroupManagementTests_GroupUpdate_Test:



Additional Inherited Members

5.68.1 Detailed Description

Test Update an existing group and make sure it can be retrieved correctly.

1. Create a valid security group.

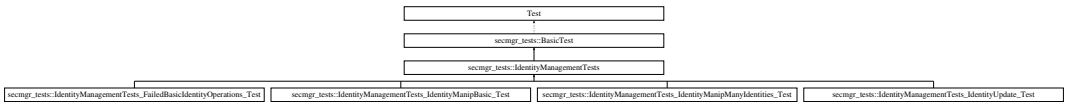
2. Store the group and make sure this is successful.
3. Retrieve the group from storage and make sure this is successful.
4. Change the name and description of the group.
5. Store the group and make sure this is successful.
6. Retrieve the group and make sure it matches the updated info.

The documentation for this class was generated from the following file:

- GroupManagementTests.cc

5.69 secmgr_tests::IdentityManagementTests Class Reference

Inheritance diagram for secmgr_tests::IdentityManagementTests:



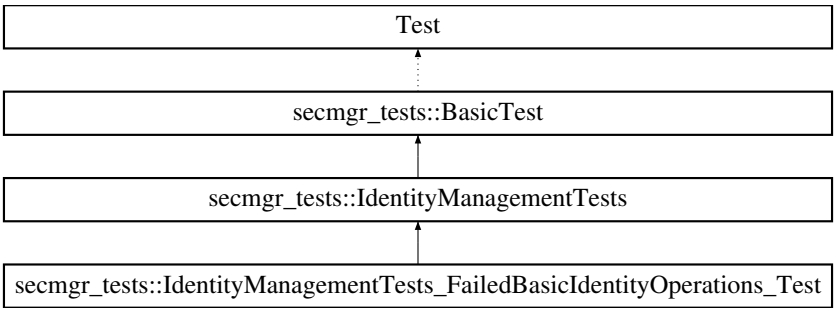
Additional Inherited Members

The documentation for this class was generated from the following file:

- IdentityManagementTests.cc

5.70 secmgr_tests::IdentityManagementTests_FailedBasicIdentityOperations_Test Class Reference

Inheritance diagram for secmgr_tests::IdentityManagementTests_FailedBasicIdentityOperations_Test:



Additional Inherited Members

5.70.1 Detailed Description

Test Retrieval and deletion of unknown identities should fail.

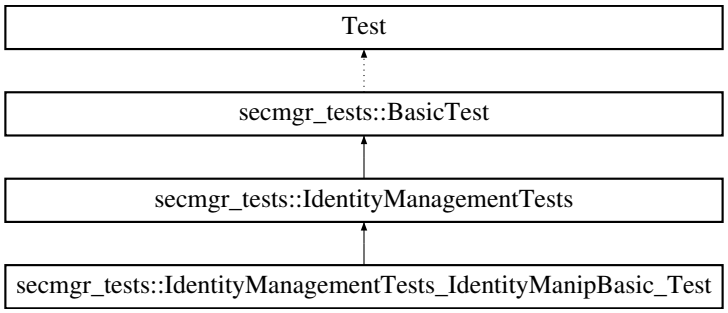
1. Try to get an unknown identity and make sure this fails.
2. Try to remove an unknown identity and make sure this fails.
3. Try to get all identities and make sure there are none.

The documentation for this class was generated from the following file:

- IdentityManagementTests.cc

5.71 secmgr_tests::IdentityManagementTests_IdentityManipBasic_Test Class Reference

Inheritance diagram for secmgr_tests::IdentityManagementTests_IdentityManipBasic_Test:



Additional Inherited Members

5.71.1 Detailed Description

Test Store, retrieve and delete an identity from storage.

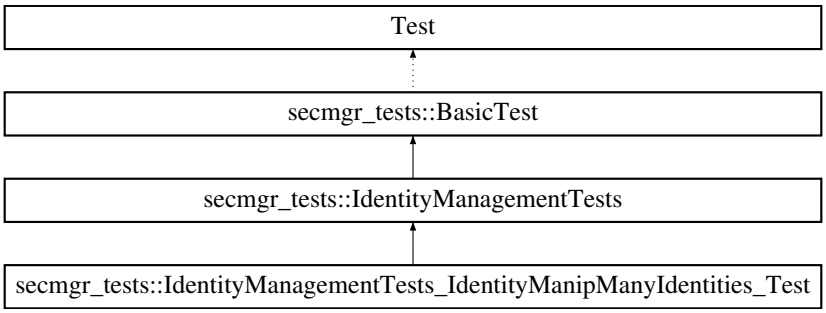
1. Define a valid IdentityInfo and store it.
2. Retrieve the identity from storage and check whether it matches the stored identity.
3. Remove the identity from storage.
4. Retrieving the identity should fail.

The documentation for this class was generated from the following file:

- IdentityManagementTests.cc

5.72 secmgr_tests::IdentityManagementTests_IdentityManipManyIdentities_Test Class Reference

Inheritance diagram for secmgr_tests::IdentityManagementTests_IdentityManipManyIdentities_Test:



Additional Inherited Members

5.72.1 Detailed Description

Test Store, retrieve and delete many identities from storage.

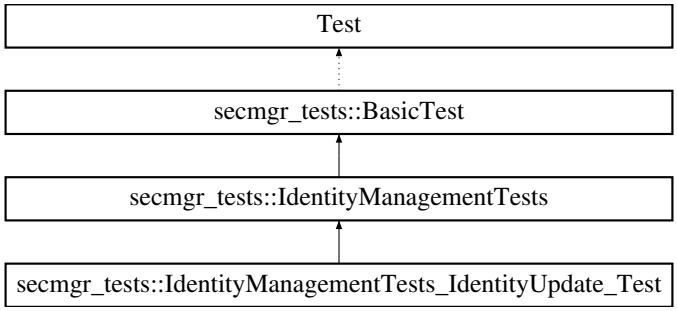
1. Define a series of identities and store them one by one.
2. Retrieve all identities from storage and count whether all have been stored correctly.
3. Remove all identities one by one from storage.
4. Retrieve all identities from storage and make sure none are returned.

The documentation for this class was generated from the following file:

- IdentityManagementTests.cc

5.73 secmgr_tests::IdentityManagementTests_IdentityUpdate_Test Class Reference

Inheritance diagram for secmgr_tests::IdentityManagementTests_IdentityUpdate_Test:



Additional Inherited Members

5.73.1 Detailed Description

Test Update an existing identity and make sure it can be retrieved correctly.

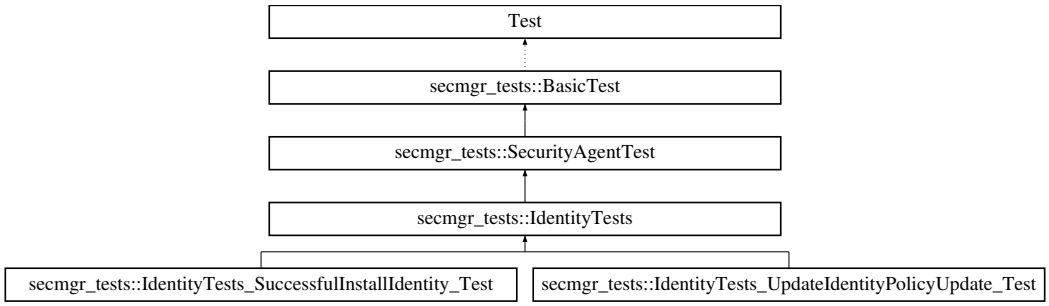
1. Create a valid identity.
2. Store the identity and make sure this is successful.
3. Retrieve the identity from storage and make sure this is successful.
4. Change the name and description of the identity.
5. Store the identity and make sure this is successful.
6. Retrieve the identity and make sure it matches the updated identity.

The documentation for this class was generated from the following file:

- IdentityManagementTests.cc

5.74 secmgr_tests::IdentityTests Class Reference

Inheritance diagram for secmgr_tests::IdentityTests:



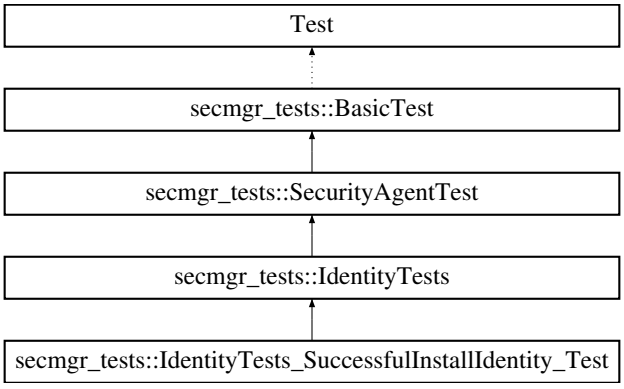
Additional Inherited Members

The documentation for this class was generated from the following file:

- IdentityTests.cc

5.75 secmgr_tests::IdentityTests_SuccessfulInstallIdentity_Test Class Reference

Inheritance diagram for secmgr_tests::IdentityTests_SuccessfulInstallIdentity_Test:



Additional Inherited Members

5.75.1 Detailed Description

Test Update the identity certificate of an application and check that it gets installed correctly.

1. Start the application.
2. Make sure the application is in a CLAIMABLE state.
3. Create and store an IdentityInfo.
4. Claim the application using the IdentityInfo.
5. Check whether the application becomes CLAIMED.
6. Create and store another IdentityInfo.
7. Update the identity certificate of the application.

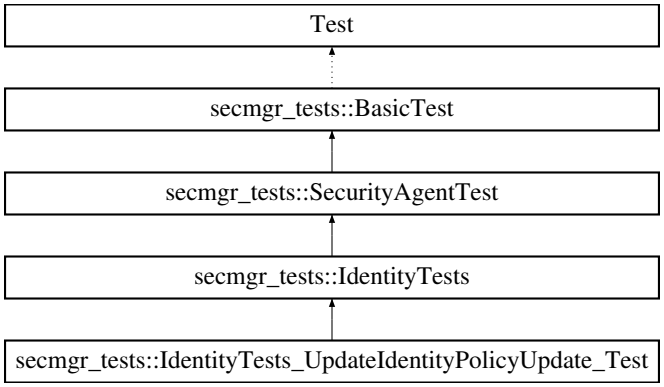
8. Wait for the updates to be completed.
9. Check whether the identity certificate was installed successfully.
10. Remove the latest identity and make sure the app is removed and that it becomes claimable again.
11. Use the original identity to claim 2 apps successfully.
12. Remove the original identity and verify that the applications are removed and they are claimable again.
13. Get all managed applications and verify that none exists.

The documentation for this class was generated from the following file:

- IdentityTests.cc

5.76 secmgr_tests::IdentityTests_UpdateIdentityPolicyUpdate_Test Class Reference

Inheritance diagram for secmgr_tests::IdentityTests_UpdateIdentityPolicyUpdate_Test:



Additional Inherited Members

5.76.1 Detailed Description

Test Verify that update identity triggers an increase of the policy version

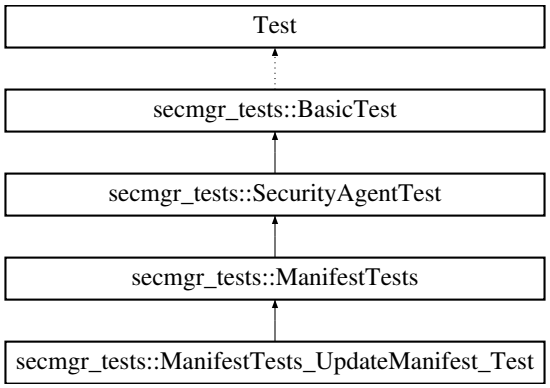
1. Start an application and make sure it's online and CLAIMABLE.
2. Successfully store an IdentityInfo instance.
3. Successfully claim the application using the IdentityInfo instance.
4. Make sure the application is online and in the CLAIMED state with no updates pending.
5. Make sure the remote identity and manifest of the application match the stored ones.
6. Update a policy on an application
7. Verify that updateIdentity is succesful.
8. Make sure updates have been completed.
9. Check that the policy version increased.

The documentation for this class was generated from the following file:

- IdentityTests.cc

5.77 secmgr_tests::ManifestTests Class Reference

Inheritance diagram for secmgr_tests::ManifestTests:



Public Member Functions

- void **GetManifest** (Manifest &mf)
- void **GetExtendedManifest** (Manifest &mf)

Public Attributes

- IdentityInfo **idInfo**

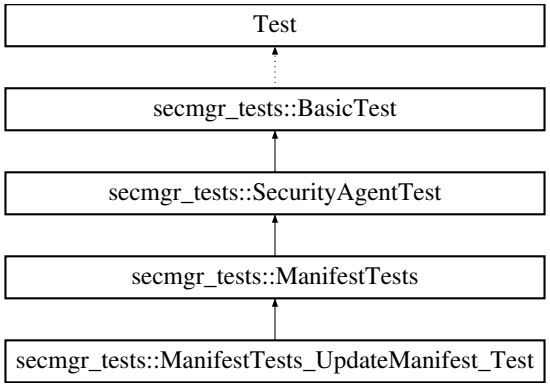
Additional Inherited Members

The documentation for this class was generated from the following file:

- ManifestTests.cc

5.78 secmgr_tests::ManifestTests_UpdateManifest_Test Class Reference

Inheritance diagram for secmgr_tests::ManifestTests_UpdateManifest_Test:



Additional Inherited Members

5.78.1 Detailed Description

Test Update the manifest of an application and check whether a ManifestUpdate event is triggered if the manifest contains additional rules.

1. Set the manifest of the application to manifest1.
2. Claim the application and check whether the manifest during claiming matches the remote manifest.
3. Set the manifest of the application to manifest2 which extends manifest1.
4. Check whether a ManifestUpdate event is triggered.
5. Update the identity certificate based on the newly requested manifest.
6. Check that no additional ManifestUpdate events are triggered.
7. Set the manifest of the application to manifest1.
8. Make sure no additional ManifestUpdate events are triggered.

The documentation for this class was generated from the following file:

- ManifestTests.cc

5.79 secmgr_tests::ManifestUtilTests Class Reference

Inheritance diagram for secmgr_tests::ManifestUtilTests:

Public Member Functions

- QStatus **GenerateManifest** (PermissionPolicy::Rule ∗∗retRules, size_t ∗count)
- void **GetManifest** (Manifest &mf)
- void **GetPermutedManifest** (Manifest &mf)
- void **GetSplitManifest** (Manifest &mf)
- void **GetExtendedManifest** (Manifest &mf)

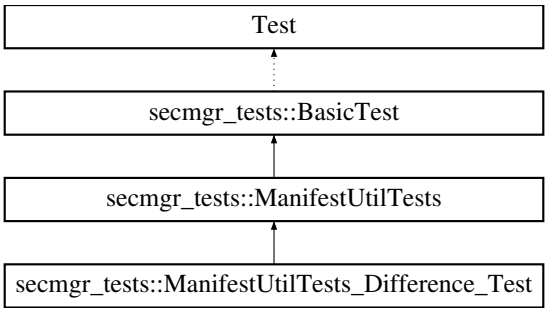
Additional Inherited Members

The documentation for this class was generated from the following file:

- ManifestUtilTests.cc

5.80 secmgr_tests::ManifestUtilTests_Difference_Test Class Reference

Inheritance diagram for secmgr_tests::ManifestUtilTests_Difference_Test:



Additional Inherited Members

5.80.1 Detailed Description

Test Verify the difference between two manifest is computed correctly.

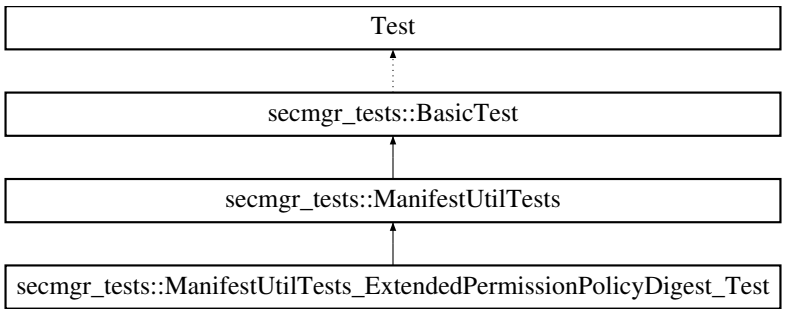
1. Create two manifests: one basic manifest, and one extending the basic manifest by adding another interface and by extending the action mask on a specific member.
2. Compute the difference between the extended manifest and the basic manifest, and check whether the outcome is as expected.
3. Compute the difference between the basic manifest and the extend manifest, and make sure it is empty.

The documentation for this class was generated from the following file:

- ManifestUtilTests.cc

5.81 secmgr_tests::ManifestUtilTests_ExtendedPermissionPolicyDigest_Test Class Reference

Inheritance diagram for secmgr_tests::ManifestUtilTests_ExtendedPermissionPolicyDigest_Test:



Additional Inherited Members

5.81.1 Detailed Description

Test Verify the PermissionPolicy's digest consistency after copy or assignment operations. Also verify the functionalities provided by the static public Util class.

1. Using a DefaultPolicyMarshaller create a PermissionPolicy (permPolicy) and get its digest.
2. Successfully create a copy PermissionPolicy (permPolicyCopy) and get its digest.

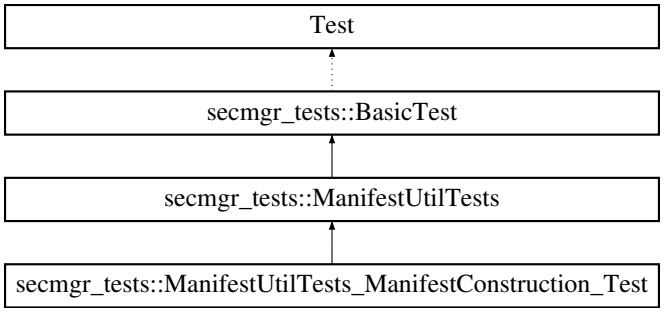
3. Compare digests from permPolicyCopy and permPolicy and make sure they match.
4. Create a PermissionPolicy (permPolicyAssignee) using the assignment operator from permPolicy and make sure its digest matches those from permPolicyCopy and permPolicy.
5. In a scoped block, initialize the Util class and get the byte-array of permPolicy successfully. Using that byteArray invoke GetPolicy on Util to create a policyFromImport successfully. Finalize the Util successfully.
6. Finally, make sure the digest of policyFromImport matches that of permPolicy.

The documentation for this class was generated from the following file:

- ManifestUtilTests.cc

5.82 secmgr_tests::ManifestUtilTests_ManifestConstruction_Test Class Reference

Inheritance diagram for secmgr_tests::ManifestUtilTests_ManifestConstruction_Test:



Additional Inherited Members

5.82.1 Detailed Description

Test Verify the construction of valid Manifest objects using the Manifest class. Also verify the provided operators and the digest matching.

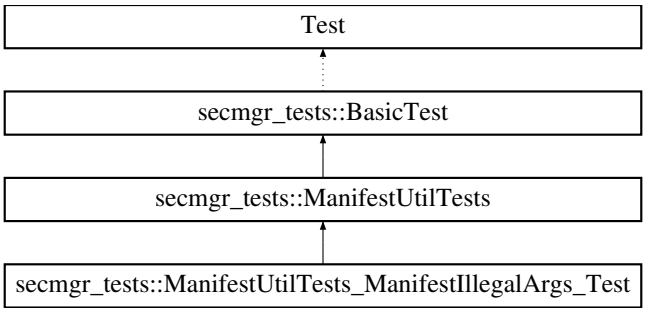
1. Create an empty manifest object and make sure that its rules and byte-array are empty. Both sizes of rules and the byte-array should be zero.
2. Create a manifest from some two generated rules (manifestFromRules) and verify that it has those exact generated rules. Also, make sure that the corresponding byte-array is not empty.
3. Repeat the previous step for a manifest created form the previous byte-array (manifestFromByteArray). Also, verify that the byte-array of manifestFromRules matches that of manifestFromByteArray.
4. Get digests of both manifestFromByteArray and manifestFromRules and make sure they match.
5. Create a copy (copyManifest) form manifestFromByteArray using the copy constructor and make sure == and != operators against manifestFromByteArray and manifestFromRules hold.
6. Create a manifestAssignee using the assignment operator form manifestFromByteArray and make sure == and != operators against manifestFromByteArray and manifestFromRules hold.
7. Get the digests from manifestAssignee, manifestFromByteArray and copyManifest and make sure they are all identical.

The documentation for this class was generated from the following file:

- ManifestUtilTests.cc

5.83 secmgr_tests::ManifestUtilTests_ManifestIllegalArgs_Test Class Reference

Inheritance diagram for secmgr_tests::ManifestUtilTests_ManifestIllegalArgs_Test:



Additional Inherited Members

5.83.1 Detailed Description

Test Verify that the Manifest class methods can handle illegal arguments.

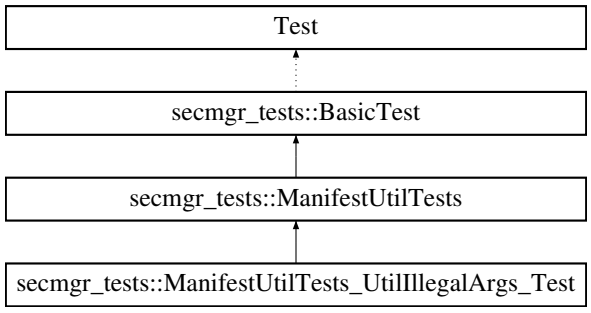
1. Try to construct a manifest from nullptr rules and/or a size <= 0 and make sure then it is equal to a manifest created with the default constructor.
2. Try to construct a manifest from nullptr byte-array and/or a size <= 0 and make sure then it is equal to a manifest created with the default constructor.
3. Call all getter functions on a manifest created with the default constructor with nullptr inputs and make sure this fails (!= ER_OK).
4. Call all setter functions on a manifest created with the default constructor with nullptr first argument and size <= 0 and make sure all will fail (!= ER_OK).
5. Call all setter functions on a manifest created with the default constructor with valid first argument and size <= 0 and make sure all will fail (!= ER_OK).

The documentation for this class was generated from the following file:

- ManifestUtilTests.cc

5.84 secmgr_tests::ManifestUtilTests_UtilIllegalArgs_Test Class Reference

Inheritance diagram for secmgr_tests::ManifestUtilTests_UtilIllegalArgs_Test:



Additional Inherited Members

5.84.1 Detailed Description

Test Verify that the Util class methods can handle illegal arguments.

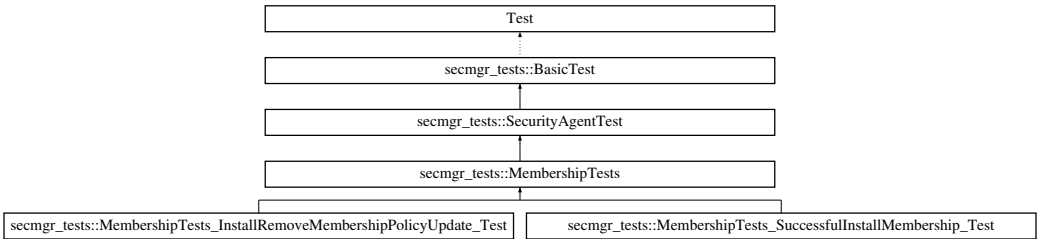
1. Init Util with a nullptr argument and make sure it fails.
2. Call all methods of Util with valid arguments and make sure they all fail (!= ER_OK).
3. Init Util with a valid busattachment and make sure it succeeds.
4. Call GetDefaultMarshaller with nullptr argument and make sure it fails (!= ER_OK).
5. Call GetPolicyByteArray with nullptr byte-array and size and make sure this fails (!= ER_OK).
6. Call GetPolicy with nullptr byte-array and/or size <= 0 and make sure this fails (!= OR_OK).
7. Call Fini on Util and make sure it succeeds (== ER_OK).

The documentation for this class was generated from the following file:

- ManifestUtilTests.cc

5.85 secmgr_tests::MembershipTests Class Reference

Inheritance diagram for secmgr_tests::MembershipTests:



Public Attributes

- IdentityInfo **idInfo**
- GroupInfo **groupInfo1**
- GroupInfo **groupInfo2**

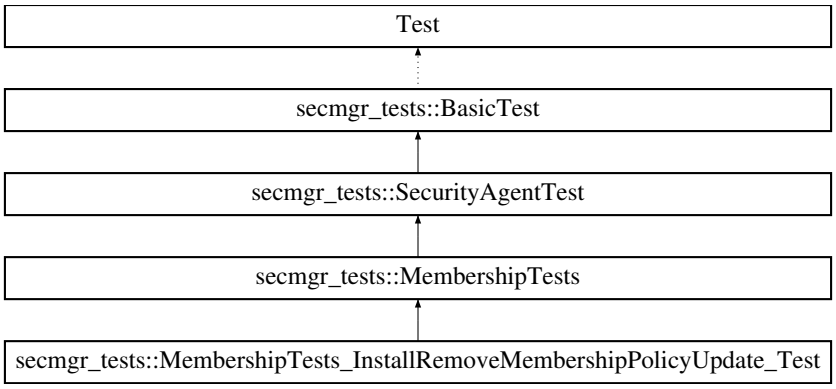
Additional Inherited Members

The documentation for this class was generated from the following file:

- MembershipTests.cc

5.86 secmgr_tests::MembershipTests_InstallRemoveMembershipPolicyUpdate_Test Class Reference

Inheritance diagram for secmgr_tests::MembershipTests_InstallRemoveMembershipPolicyUpdate_Test:



Additional Inherited Members

5.86.1 Detailed Description

Test Verify that installing and removing a membership triggers an increase in the policy version

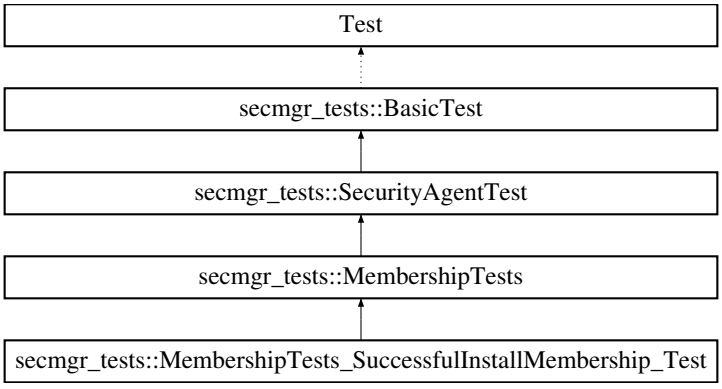
1. Start an application and make sure it's online and CLAIMABLE.
2. Successfully store an IdentityInfo instance.
3. Successfully claim the application using the IdentityInfo instance.
4. Make sure the application is online and in the CLAIMED state with no updates pending.
5. Make sure the remote identity and manifest of the application match the stored ones.
6. Update a policy on an application
7. Verify that installing of membership using groupInfo1 is successful.
8. Make sure updates have been completed.
9. Check that the policy version increased.
10. Verify that removing of membership using groupInfo1 is successful.
11. Make sure updates have been completed.
12. Check that the policy version increased again.

The documentation for this class was generated from the following file:

- MembershipTests.cc

5.87 secmgr_tests::MembershipTests_SuccessfulInstallMembership_Test Class Reference

Inheritance diagram for secmgr_tests::MembershipTests_SuccessfulInstallMembership_Test:



Additional Inherited Members

5.87.1 Detailed Description

Test Verify the ability to install several memberships based on different GroupInfo instances.

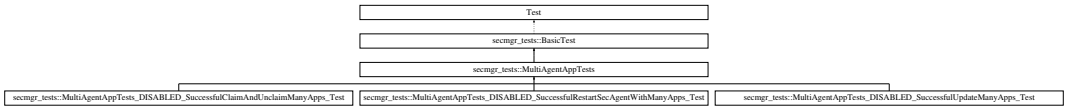
1. Store a couple of different GroupInfo instances; groupInfo1 and groupInfo2 in persistency.
2. Start an application and make sure it's online and CLAIMABLE.
3. Try to install and remove a membership using the lately announced application and make sure this fails.
4. Successfully store an IdentityInfo instance.
5. Successfully claim the application using the IdentityInfo instance.
6. Make sure the application is online and in the CLAIMED state with no updates pending.
7. Make sure the remote identity and manifest of the application match the stored ones.
8. Verify that installing of membership using groupInfo1 is successful.
9. Make sure updates have been completed.
10. Repeat the previous 2 steps for groupInfo2.
11. Verify that removal of membership using groupInfo1 is successful.
12. Make sure updates have been completed.
13. Repeat the previous 2 steps for groupInfo2.
14. Install memberships for both groupInfo1 and groupInfo2 successfully.
15. Verify that deleting groupInfo1 and groupInfo2 will result in syncing the app again and in removing the memberships associated.
16. Repeat the previous step but verify the removal of memberships associated immediately after the deletion of each group.

The documentation for this class was generated from the following file:

- MembershipTests.cc

5.88 secmgr_tests::MultiAgentAppTests Class Reference

Inheritance diagram for secmgr_tests::MultiAgentAppTests:



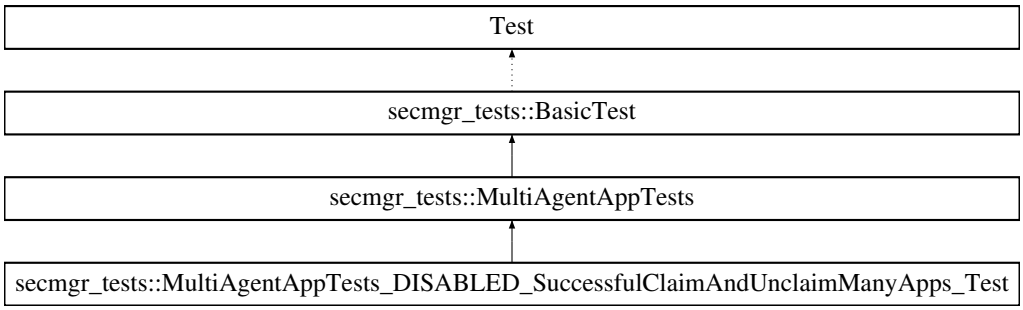
Additional Inherited Members

The documentation for this class was generated from the following file:

- MultiAgentAppTests.cc

5.89 secmgr_tests::MultiAgentAppTests_DISABLED_SuccessfulClaimAndUnclaimManyApps_Test Class Reference

Inheritance diagram for secmgr_tests::MultiAgentAppTests_DISABLED_SuccessfulClaimAndUnclaimManyApps_Test:



Additional Inherited Members

5.89.1 Detailed Description

Test Verify that where multi-agents are active, the applications needing to be claimed and be managed are dealt with consistently.

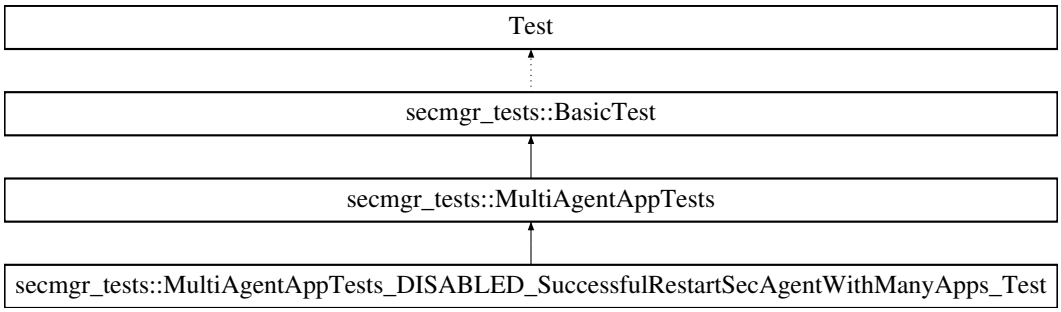
1. Start an number of security agents which share the same storage.
2. Start a number of applications and make sure that all agents see those applications in the CLAIMABLE state.
3. Randomly, let the security agents start claiming random claimable applications.
4. Verify that all applications have been claimed successfully.
5. Randomly, let the security agents start unclaiming random claimed applications.
6. Verify that all applications have changed state to CLAIMABLE

The documentation for this class was generated from the following file:

- MultiAgentAppTests.cc

5.90 secmgr_tests::MultiAgentAppTests_DISABLED_SuccessfulRestartSecAgentWithManyApps_Test Class Reference

Inheritance diagram for secmgr_tests::MultiAgentAppTests_DISABLED_SuccessfulRestartSecAgentWithManyApps_Test:



Additional Inherited Members

5.90.1 Detailed Description

Test Verify that a restarted Security Agent (SA) after claiming many applications will maintain a consistent view.

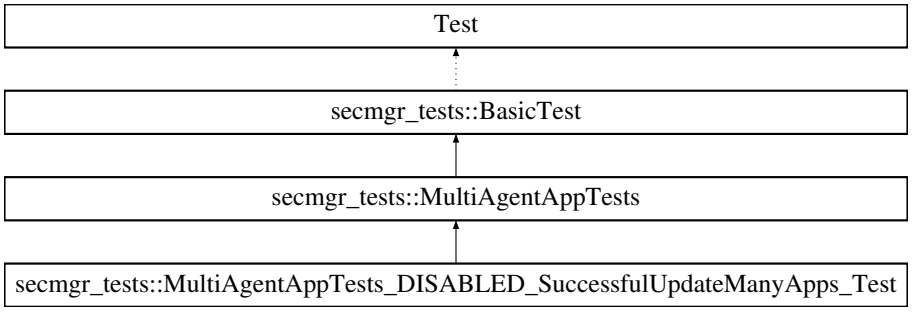
1. Start a number of applications and make sure they are claimable.
2. Start an SA and verify that it could see all the online applications in a CLAIMABLE state.
3. Ask the SA to start claiming (at no specific order) all claimable applications.
4. Verify that all applications have been claimed; state change to CLAIMED.
5. Ask the SA to install/update all claimed applications with a predefined membership certificate and policy.
6. Kill the SA.
7. Start the SA again and make sure that all previously claimed applications are in a consistent online state and no sync errors will have occurred.
8. Kill all applications.
9. Verify that the SA considers all applications as offline.

The documentation for this class was generated from the following file:

- MultiAgentAppTests.cc

5.91 secmgr_tests::MultiAgentAppTests_DISABLED_SuccessfulUpdateManyApps_Test Class Reference

Inheritance diagram for secmgr_tests::MultiAgentAppTests_DISABLED_SuccessfulUpdateManyApps_Test:



Additional Inherited Members

5.91.1 Detailed Description

Test Verify that where multi-agents are active, claimed applications can be updated correctly after a restart.

1. Start an number of security agents which share the same storage.
2. Start a number of applications and make sure that all agents see those applications in the CLAIMABLE state.
3. Claim successfully all applications...could be using one security agent.
4. Verify that all applications have been claimed successfully.
5. Stop all claimed applications.
6. Make sure all security agents see all applications as off-line (i.e., no busName).
7. Update all applications with predefined membership and policy using the available security agents randomly.
8. Verify that storage has the predefined membership and policy for all applications.
9. Restart all the applications.
10. Verify that all applications have been updated correctly and that no sync errors have been encountered.

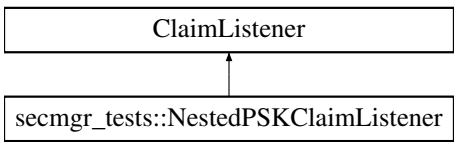
11. Verify all applications are in the CLAIMED state.

The documentation for this class was generated from the following file:

- MultiAgentAppTests.cc

5.92 secmgr_tests::NestedPSKClaimListener Class Reference

Inheritance diagram for secmgr_tests::NestedPSKClaimListener:



Public Member Functions

- **NestedPSKClaimListener** (vector< OnlineApplication > &_apps, vector< shared_ptr< TestApplication > > &_testapps, IdentityInfo &_idInfo, shared_ptr< SecurityAgent > &_secMgr)
- QStatus **ApproveManifestAndSelectSessionType** (ClaimContext &ctx)

Public Attributes

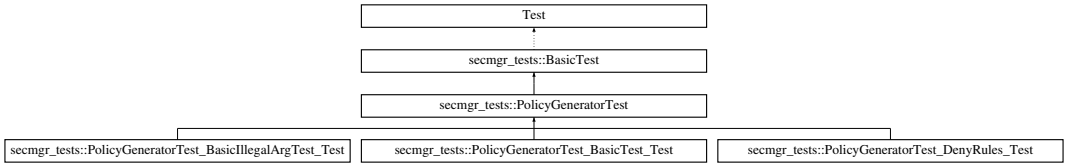
- IdentityInfo **idInfo**
- vector< OnlineApplication > **apps**
- vector< shared_ptr < TestApplication > > **testapps**
- shared_ptr< SecurityAgent > **secMgr**

The documentation for this class was generated from the following file:

- ClaimingTests.cc

5.93 secmgr_tests::PolicyGeneratorTest Class Reference

Inheritance diagram for secmgr_tests::PolicyGeneratorTest:



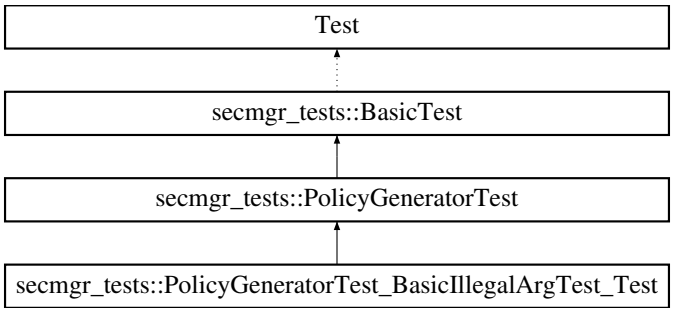
Additional Inherited Members

The documentation for this class was generated from the following file:

- PolicyGeneratorTests.cc

5.94 secmgr_tests::PolicyGeneratorTest_BasicIllegalArgTest_Test Class Reference

Inheritance diagram for secmgr_tests::PolicyGeneratorTest_BasicIllegalArgTest_Test:



Additional Inherited Members

5.94.1 Detailed Description

Test Basic test for illegal argument in policy generator.

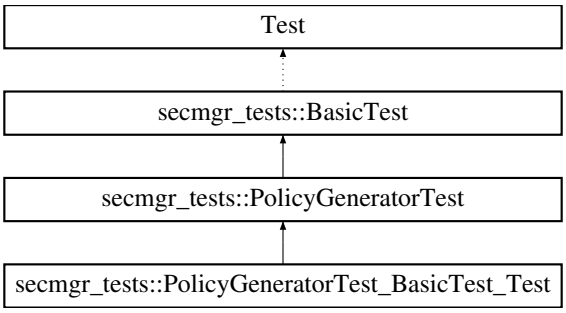
1. Create an empty vector of GroupInfo; groupInfoEmptyVec
2. Use the existing PolicyGenerator instance (pg) to get a DefaultPolicy using the groupInfoEmptyVec and make sure this does not fail but return a default policy with one admin rule.

The documentation for this class was generated from the following file:

- PolicyGeneratorTests.cc

5.95 secmgr_tests::PolicyGeneratorTest_BasicTest_Test Class Reference

Inheritance diagram for secmgr_tests::PolicyGeneratorTest_BasicTest_Test:



Additional Inherited Members

5.95.1 Detailed Description

Test Basic test for the sample policy generator.

1. Create a security group and store it.
2. Generate a policy for this group.
3. Make sure this policy has two ACLs (including one for the admin group).

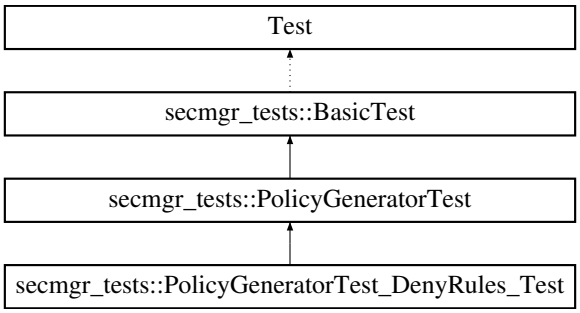
4. Create another security group and store it.
5. Generate another policy for both groups.
6. Make sure this policy has three ACLs (including one for the admin group).

The documentation for this class was generated from the following file:

- PolicyGeneratorTests.cc

5.96 secmgr_tests::PolicyGeneratorTest_DenyRules_Test Class Reference

Inheritance diagram for secmgr_tests::PolicyGeneratorTest_DenyRules_Test:



Additional Inherited Members

5.96.1 Detailed Description

Test Validate the generation of a policy with deny rules.

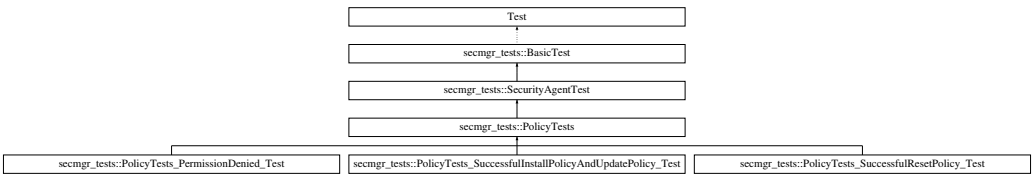
1. Create a policy generator and add a random application to its blacklist.
2. Generate a default policy for an empty vector of groups.
3. Check that the resulting policy has 2 ACLs.
4. Check that the resulting policy only contains valid deny rules.

The documentation for this class was generated from the following file:

- PolicyGeneratorTests.cc

5.97 secmgr_tests::PolicyTests Class Reference

Inheritance diagram for secmgr_tests::PolicyTests:



Public Attributes

- IdentityInfo **idInfo**
- GUID128 **groupGUID**
- GUID128 **groupGUID2**
- PermissionPolicy **policy**
- PermissionPolicy **policy2**

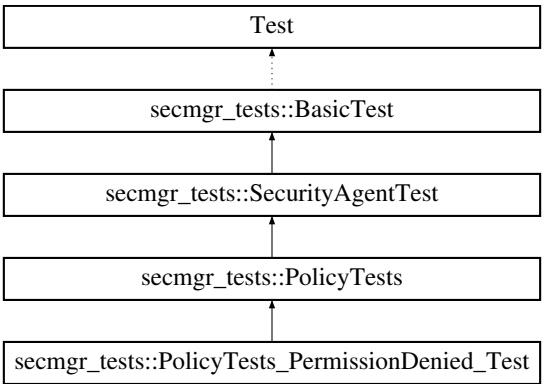
Additional Inherited Members

The documentation for this class was generated from the following file:

- PolicyTests.cc

5.98 secmgr_tests::PolicyTests_PermissionDenied_Test Class Reference

Inheritance diagram for secmgr_tests::PolicyTests_PermissionDenied_Test:



Additional Inherited Members

5.98.1 Detailed Description

Test Verify the the security agent can handle permission denied response.

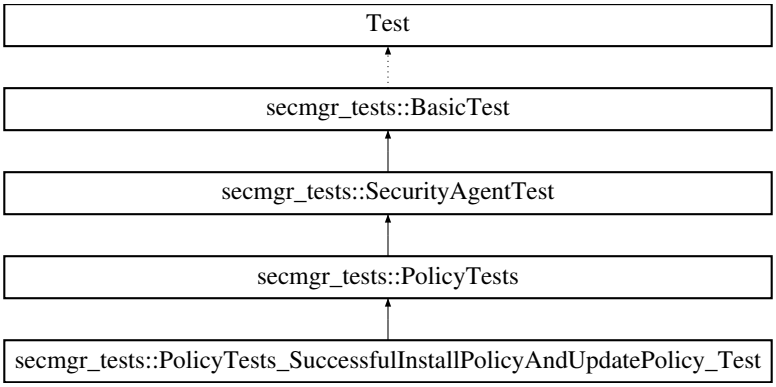
1. Start the application and make sure it's claimable.
2. Claim the application successfully.
3. Install a policy that does NOT contain the admin group rule.
4. Check whether the application is in SYNC_PENDING state.
5. Make sure that at least one sync error is triggered.

The documentation for this class was generated from the following file:

- PolicyTests.cc

5.99 secmgr_tests::PolicyTests_SuccessfulInstallPolicyAndUpdatePolicy_Test Class Reference

Inheritance diagram for secmgr_tests::PolicyTests_SuccessfulInstallPolicyAndUpdatePolicy_Test:



Additional Inherited Members

5.99.1 Detailed Description

Test Update the policy of an application and check whether it is updated correctly.

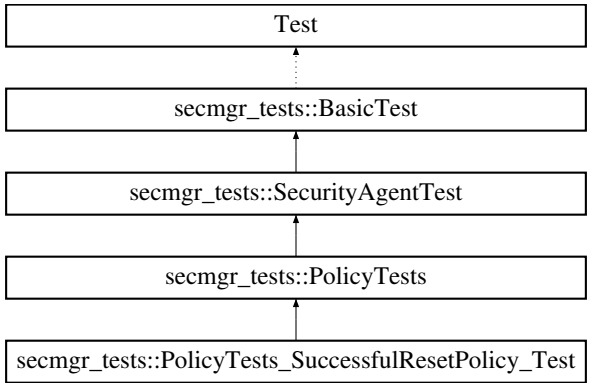
1. Start the application.
2. Installing and retrieving the policy before claiming should fail.
3. Make sure the application is in a CLAIMABLE state.
4. Create and store an IdentityInfo.
5. Claim the application using the IdentityInfo.
6. Accept the manifest of the application.
7. Check whether the application becomes CLAIMED.
8. Make sure the retrieval of the policy returns ER_END_OF_DATA.
9. Update the policy.
10. Wait for updates to complete.
11. Update the policy again.
12. Check whether the remote policy is equal to the installed policy.
13. Check whether the remote policy is equal to the policy that can be retrieved from storage.
14. Wait for updates to complete.
15. Check whether the remote policy is equal to the installed policy.
16. Check whether the remote policy is equal to the policy that can be retrieved from storage.
17. Try to install a newer policy (version 100) and verify it was successful
18. Try to install an older policy (version 1) and verify this fails
19. Try to install a default policy of (version 0) and verify this was successful.
20. Get the persisted policy and make sure its version is (version 100 +1)

The documentation for this class was generated from the following file:

- PolicyTests.cc

5.100 secmgr_tests::PolicyTests_SuccessfulResetPolicy_Test Class Reference

Inheritance diagram for secmgr_tests::PolicyTests_SuccessfulResetPolicy_Test:



Additional Inherited Members

5.100.1 Detailed Description

Test Verify resetting the policy of an application succeeds.

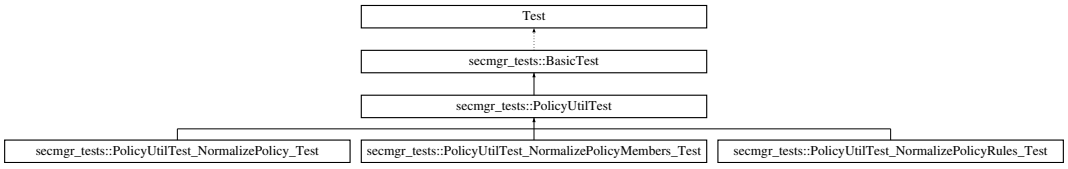
1. Start the application and make sure it's claimable.
2. Claim the application successfully.
3. Check the default policy.
4. Install a different policy and wait until updates have been completed.
5. Check whether the policy was installed successfully.
6. Reset the policy and wait until updates have been completed.
7. Check the default policy.

The documentation for this class was generated from the following file:

- PolicyTests.cc

5.101 secmgr_tests::PolicyUtilTest Class Reference

Inheritance diagram for secmgr_tests::PolicyUtilTest:



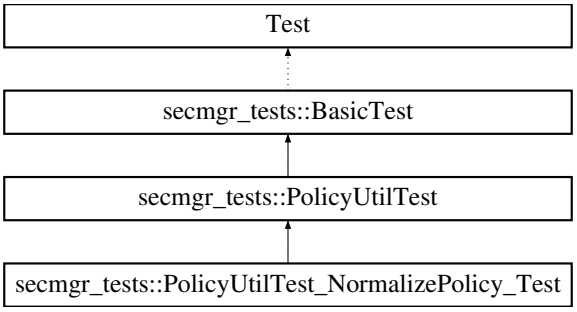
Additional Inherited Members

The documentation for this class was generated from the following file:

- PolicyUtilTests.cc

5.102 secmgr_tests::PolicyUtilTest_NormalizePolicy_Test Class Reference

Inheritance diagram for secmgr_tests::PolicyUtilTest_NormalizePolicy_Test:



Additional Inherited Members

5.102.1 Detailed Description

Test Normalize a policy with partially matching rules, and check whether the resulting policy matches the expected outcome.

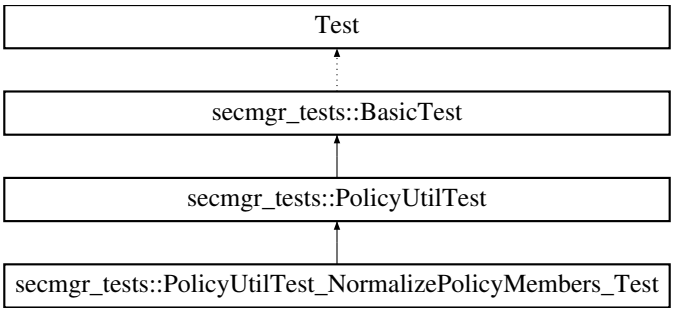
1. Create a rule with a specific InterfaceName, and one Members with a specific MemberName, and with the ActionMask set to ACTION_OBSERVE.
2. Create another rule with the same InterfaceName, and two Members. One member with a matching name to the previous rule, and one with a different name, each having the ActionMask set to ACTION_-MODIFY.
3. Create a third rule with a different InterfaceName but with a member of a matching MemberName. The ActionMask should be set to ACTION_PROVIDE.
   (a) Add those rules to a policy.
   (b) Normalize the policy.
4. Verify that the normalized policy has two rules (one for each InterfaceName).
5. Verify that the rule with the matching InterfaceName contains two members and verify that the Action-Masks are collapsed successfully.
6. Verify that the rule with the unique InterfaceName has only one member and verify the resulting Action-Mask.

The documentation for this class was generated from the following file:

- PolicyUtilTests.cc

5.103 secmgr_tests::PolicyUtilTest_NormalizePolicyMembers_Test Class Reference

Inheritance diagram for secmgr_tests::PolicyUtilTest_NormalizePolicyMembers_Test:



Additional Inherited Members

5.103.1 Detailed Description

Test Normalize a rule with matching members, and see whether the resulting action mask corresponds to the OR value of the ActionMasks of all individual members.

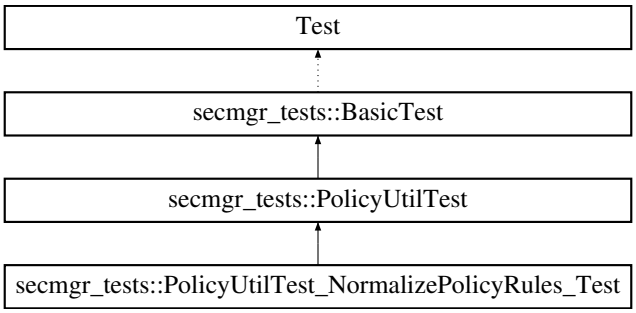
1. Create a rule with three matching members with different action masks: one with ACTION_PROVIDE, one with ACTION_OBSERVE and one with ACTION_MODIFY.
2. Add this rule to a policy.
3. Normalize this policy.
4. Verify that the normalized policy has only one member.
5. Verify that this member has a full ActionMask.

The documentation for this class was generated from the following file:

- PolicyUtilTests.cc

5.104 secmgr_tests::PolicyUtilTest_NormalizePolicyRules_Test Class Reference

Inheritance diagram for secmgr_tests::PolicyUtilTest_NormalizePolicyRules_Test:



Additional Inherited Members

5.104.1 Detailed Description

Test Normalize a policy with matching rules and members, and see whether the resulting action mask corresponds to the OR of the ActionMasks of all members.

1. Create a rule with two matching members with different action masks: one with ACTION_OBSERVE and one with ACTION_MODIFY.

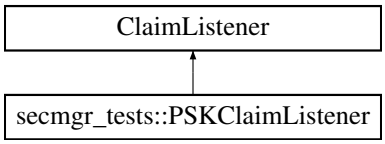
2. Create another rule that matches the previous members, but with a different action mask ACTION_PROVIDE.
3. Add those rules to a policy.
4. Normalize this policy.
5. Verify that the normalized policy has only one rule.
6. Verify that the member of this rule has a full ActionMask.

The documentation for this class was generated from the following file:

- PolicyUtilTests.cc

5.105 secmgr_tests::PSKClaimListener Class Reference

Inheritance diagram for secmgr_tests::PSKClaimListener:



Public Member Functions

- **PSKClaimListener** (const GUID128 &psk)
- QStatus **ApproveManifestAndSelectSessionType** (ClaimContext &ctx)

Public Attributes

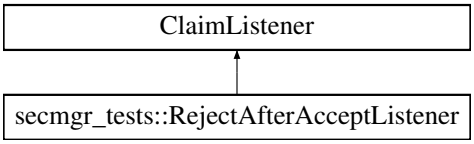
- GUID128 **localPsk**

The documentation for this class was generated from the following file:

- ClaimingTests.cc

5.106 secmgr_tests::RejectAfterAcceptListener Class Reference

Inheritance diagram for secmgr_tests::RejectAfterAcceptListener:



Public Member Functions

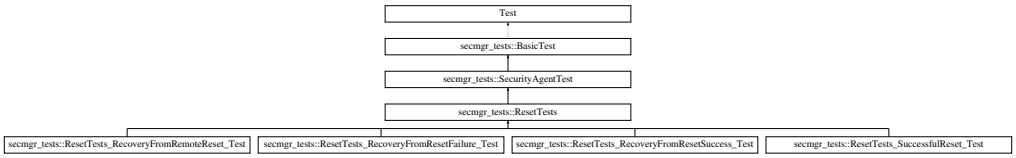
- **RejectAfterAcceptListener** (const shared_ptr< SecurityAgent > &_secMgr)
- QStatus **ApproveManifestAndSelectSessionType** (ClaimContext &ctx)

The documentation for this class was generated from the following file:

- ClaimingTests.cc

5.107 secmgr_tests::ResetTests Class Reference

Inheritance diagram for secmgr_tests::ResetTests:



Public Member Functions

- shared_ptr< AgentCAStorage > & **GetAgentCAStorage** ()

Public Attributes

- shared_ptr< FailingStorageWrapper > **wrappedCA**

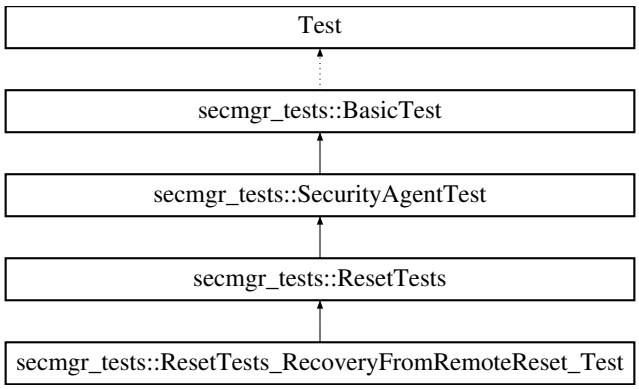
Additional Inherited Members

The documentation for this class was generated from the following file:

- ResetTests.cc

5.108 secmgr_tests::ResetTests_RecoveryFromRemoteReset_Test Class Reference

Inheritance diagram for secmgr_tests::ResetTests_RecoveryFromRemoteReset_Test:



Additional Inherited Members

5.108.1 Detailed Description

Test Discovery of an application that has been remotely reset, should result in a sync error and should be reclaimable after removing it from storage.

1. Start a test application and claim it.
2. Reset the application behind the back of the security manager.
3. Check that the security manager reports a SYNC_ER_UNEXPECTED_STATE.

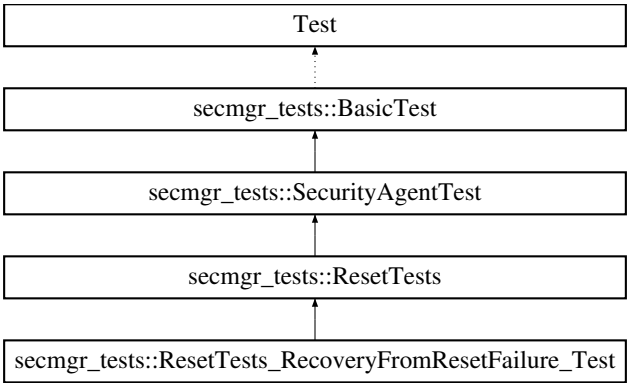
4. Check that reclaiming the application would fail.
5. Forcibly remove the application from storage.
6. Check that reclaiming the application now succeeds.

The documentation for this class was generated from the following file:

- ResetTests.cc

5.109 secmgr_tests::ResetTests_RecoveryFromResetFailure_Test Class Reference

Inheritance diagram for secmgr_tests::ResetTests_RecoveryFromResetFailure_Test:



Additional Inherited Members

5.109.1 Detailed Description

Test Recovery from failure of notifying the CA of failure to reset an application should be graceful.

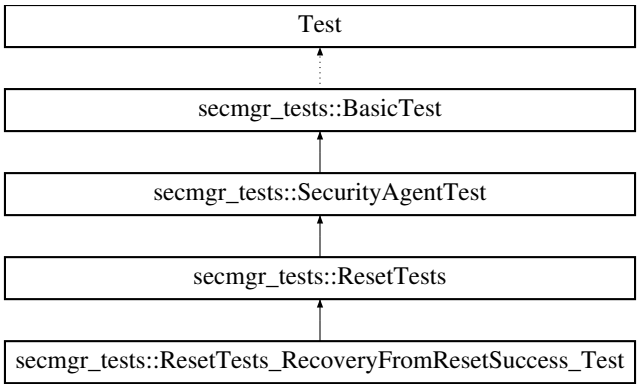
1. Start a test application and claim it.
2. Make sure remote reset fails.
3. Stop the application.
4. Make sure the UpdatesCompleted to storage fails.
5. Reset the application and check that this succeeds.
6. Restart the test application and make sure it is removed from storage.

The documentation for this class was generated from the following file:

- ResetTests.cc

5.110 secmgr_tests::ResetTests_RecoveryFromResetSuccess_Test Class Reference

Inheritance diagram for secmgr_tests::ResetTests_RecoveryFromResetSuccess_Test:



Additional Inherited Members

5.110.1 Detailed Description

Test Recovery from failure of notifying the CA of successful resetting an application should be graceful.

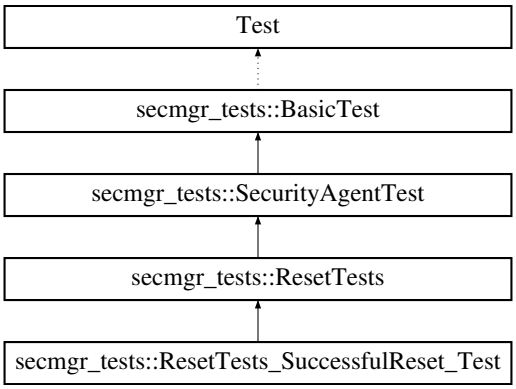
1. Start a test application and claim it.
2. Make sure the UpdatesCompleted to storage fails.
3. Reset the application and check that this succeeds.
4. Restart the test application and make sure it is removed from storage.

The documentation for this class was generated from the following file:

- ResetTests.cc

5.111 secmgr_tests::ResetTests_SuccessfulReset_Test Class Reference

Inheritance diagram for secmgr_tests::ResetTests_SuccessfulReset_Test:



Additional Inherited Members

5.111.1 Detailed Description

Test Reset an application and make sure it becomes CLAIMABLE again.

1. Start the application.

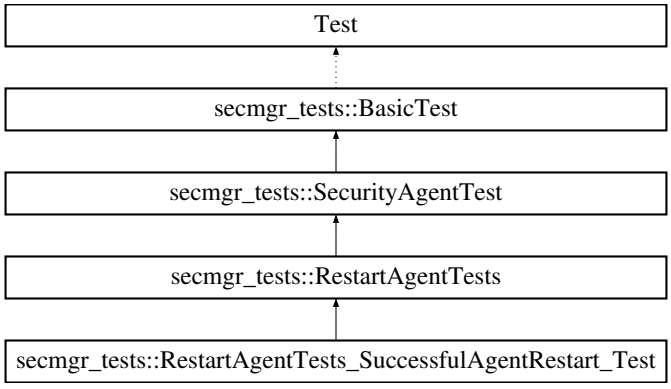
2. Make sure the application is in a CLAIMABLE state.
3. Create and store an IdentityInfo.
4. Claim the application using the IdentityInfo.
5. Accept the manifest of the application.
6. Check whether the application becomes CLAIMED.
7. Remove the application from storage.
8. Check whether it becomes CLAIMABLE again.
9. Claim the application again.
10. Check whether it becomes CLAIMED again.

The documentation for this class was generated from the following file:

- ResetTests.cc

5.112 secmgr_tests::RestartAgentTests Class Reference

Inheritance diagram for secmgr_tests::RestartAgentTests:



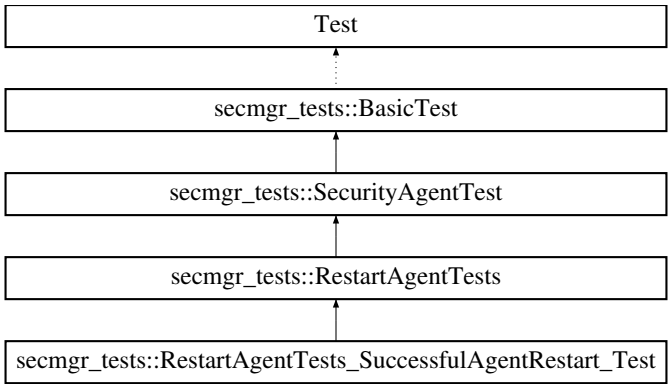
Additional Inherited Members

The documentation for this class was generated from the following file:

- RestartAgentTests.cc

5.113 secmgr_tests::RestartAgentTests_SuccessfulAgentRestart_Test Class Reference

Inheritance diagram for secmgr_tests::RestartAgentTests_SuccessfulAgentRestart_Test:



Additional Inherited Members

5.113.1 Detailed Description

Test Verify that the agent can restart and maintain a consistent view on the online applications. // See AS-1634 for the number of applications issue

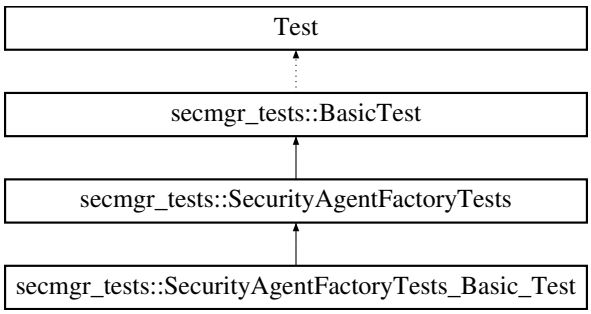
1. Start an even X number of applications and make sure they are claimable.
2. Using a dedicated IdtityInfo (per application) and a default manifest, claim an X/2 (claimedApps) of the applications.
3. Verify that claimedApps are in a CLAIMED state and that the other X/2 (claimableApps) are still in a CLAIMABLE state.
4. Delete the security agent instance that claimed the claimedApps.
5. Create a new security agent which uses the same keystore and CAStorage.
6. Verify that all online applications are in a consistent online application state from the security agent perspective.

The documentation for this class was generated from the following file:

- RestartAgentTests.cc

5.114 secmgr_tests::SecurityAgentFactoryTests Class Reference

Inheritance diagram for secmgr_tests::SecurityAgentFactoryTests:



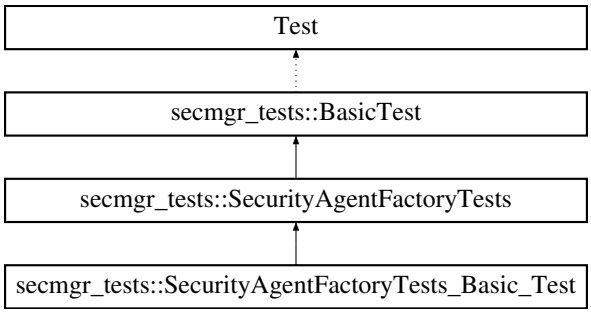
Additional Inherited Members

The documentation for this class was generated from the following file:

- SecurityAgentFactoryTests.cc

5.115 secmgr_tests::SecurityAgentFactoryTests_Basic_Test Class Reference

Inheritance diagram for secmgr_tests::SecurityAgentFactoryTests_Basic_Test:



Additional Inherited Members

5.115.1 Detailed Description

Test Ensure that the security agent factory can return a valid security agent.

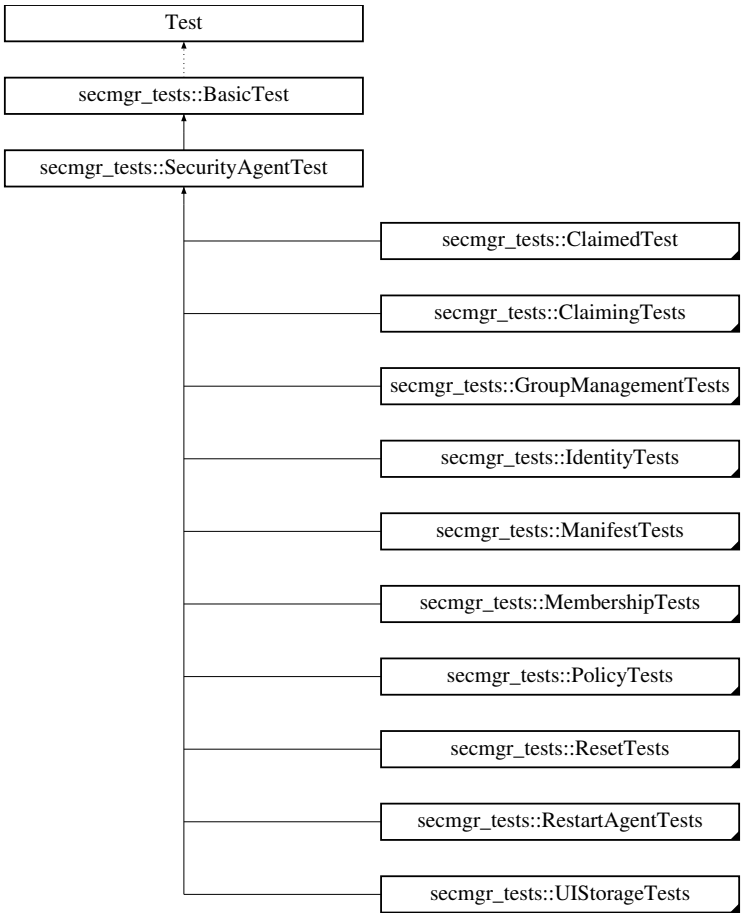
1. Get a security agent factory instance.
2. Get a security agent using a connected bus attachment.
3. Validate the security agent returned
4. Get a security agent with using a null bus attachment parameter.
5. Validate the security agent returned.

The documentation for this class was generated from the following file:

- SecurityAgentFactoryTests.cc

5.116 secmgr_tests::SecurityAgentTest Class Reference

Inheritance diagram for secmgr_tests::SecurityAgentTest:



Public Member Functions

- void **SetUp** ()

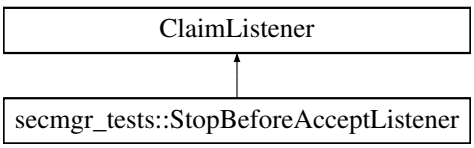
Additional Inherited Members

The documentation for this class was generated from the following file:

- TestUtil.h

5.117 secmgr_tests::StopBeforeAcceptListener Class Reference

Inheritance diagram for secmgr_tests::StopBeforeAcceptListener:



Public Member Functions

- **StopBeforeAcceptListener** (TestApplication &_testApp)

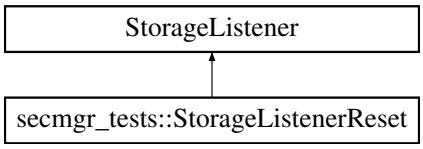
- QStatus **ApproveManifestAndSelectSessionType** (ClaimContext &ctx)

The documentation for this class was generated from the following file:

- ClaimingTests.cc

5.118 secmgr_tests::StorageListenerReset Class Reference

Inheritance diagram for secmgr_tests::StorageListenerReset:



Public Member Functions

- void **OnPendingChanges** (vector< Application > &apps)
- void **OnPendingChangesCompleted** (vector< Application > &apps)
- void **OnStorageReset** ()
- bool **WaitForStorageReset** ()

Public Attributes

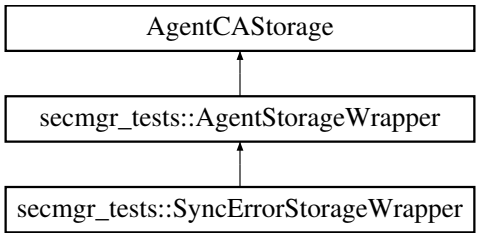
- bool **storageReset**

The documentation for this class was generated from the following file:

- UIStorageTests.cc

5.119 secmgr_tests::SyncErrorStorageWrapper Class Reference

Inheritance diagram for secmgr_tests::SyncErrorStorageWrapper:



Public Member Functions

- **SyncErrorStorageWrapper** (shared_ptr< AgentCAStorage > &_ca, shared_ptr< UIStorage > &_storage)
- QStatus **StartUpdates** (Application &app, uint64_t &updateID)
- QStatus **GetPolicy** (const Application &app, PermissionPolicy &_policy) const
- QStatus **GetIdentityCertificatesAndManifest** (const Application &app, IdentityCertificateChain &identityCertificates, Manifest &_manifest) const

- QStatus **GetMembershipCertificates** (const Application &app, vector< MembershipCertificateChain > &certs) const
- void **SetPolicy** (PermissionPolicy _policy)
- void **UnsetPolicy** ()
- void **SetManifest** (Manifest _manifest)
- void **UnsetManifest** ()

Public Attributes

- bool **failOnStartUpdates**
- bool **returnEmptyMembershipCert**

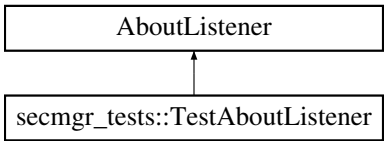
Additional Inherited Members

The documentation for this class was generated from the following file:

- ApplicationUpdaterTests.cc

5.120 secmgr_tests::TestAboutListener Class Reference

Inheritance diagram for secmgr_tests::TestAboutListener:

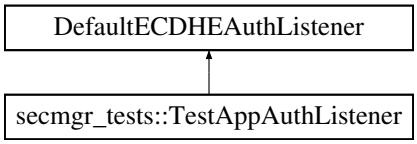


The documentation for this class was generated from the following file:

- TestUtil.h

5.121 secmgr_tests::TestAppAuthListener Class Reference

Inheritance diagram for secmgr_tests::TestAppAuthListener:



Public Member Functions

- **TestAppAuthListener** (GUID128 &psk)
- void **AuthenticationComplete** (const char *authMechanism, const char *peerName, bool success)

Public Attributes

- string **lastAuthMechanism**

The documentation for this class was generated from the following file:

- TestApplication.h

5.122 secmgr_tests::TestApplication Class Reference

Public Member Functions

- TestApplication (string _appName="secmgrctest")
- QStatus Start ()
- QStatus SetManifest (const Manifest &manifest)
- QStatus AnnounceManifest ()
- QStatus UpdateManifest (const Manifest &manifest)
- QStatus SetApplicationState (const PermissionConfigurator::ApplicationState state)
- void Reset ()
- QStatus **Stop** ()
- QStatus **SetClaimByPSK** ()
- const GUID128 & **GetPsk** () const
- const string **GetBusName** () const
- const string & **GetLastAuthMechanism** ()

5.122.1 Constructor & Destructor Documentation

5.122.1.1 secmgr_tests::TestApplication::TestApplication (string *_appName* = "secmgrctest")

Creates a new TestApplication.

5.122.2 Member Function Documentation

5.122.2.1 QStatus secmgr_tests::TestApplication::AnnounceManifest ( )

Sets the manifest on the BusAttachment.

5.122.2.2 void secmgr_tests::TestApplication::Reset ( )

Resets the keystore of this TestApplication.

5.122.2.3 QStatus secmgr_tests::TestApplication::SetApplicationState (const PermissionConfigurator::ApplicationState *state*)

Sets the application state as permitted by PermissionConfigurator

5.122.2.4 QStatus secmgr_tests::TestApplication::SetManifest (const Manifest & *manifest*)

Sets the manifest of this TestApplication.

5.122.2.5 QStatus secmgr_tests::TestApplication::Start ( )

Starts this TestApplication.

5.122.2.6 QStatus secmgr_tests::TestApplication::UpdateManifest (const Manifest & *manifest*)

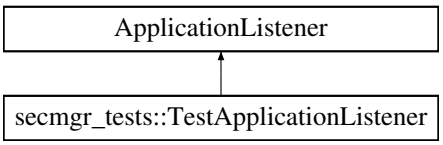
Updates the manifest of this TestApplication.

The documentation for this class was generated from the following files:

- TestApplication.h
- TestApplication.cc

5.123 secmgr_tests::TestApplicationListener Class Reference

Inheritance diagram for secmgr_tests::TestApplicationListener:



Public Member Functions

- **TestApplicationListener** (Condition &_sem, Mutex &_lock, Condition &_errorSem, Mutex &_errorLock, Condition &_manifestSem, Mutex &_manifestLock)

Public Attributes

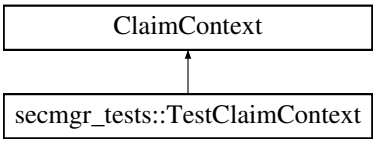
- vector< OnlineApplication > **events**
- vector< SyncError > **syncErrors**
- vector< ManifestUpdate > **manifestUpdates**

The documentation for this class was generated from the following files:

- TestUtil.h
- TestUtil.cc

5.124 secmgr_tests::TestClaimContext Class Reference

Inheritance diagram for secmgr_tests::TestClaimContext:



Public Member Functions

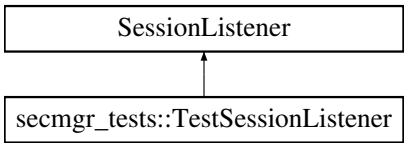
- **TestClaimContext** (const OnlineApplication &_app, const Manifest &_mnf, const PermissionConfigurator::-ClaimCapabilities _capabilities, const PermissionConfigurator::ClaimCapabilityAdditionalInfo _capInfo)
- **TestClaimContext** (const TestClaimContext &other, const PermissionConfigurator::ClaimCapabilities _-capabilities)
- QStatus **SetPreSharedKey** (const uint8_t *psk, size_t pskSize)

The documentation for this class was generated from the following file:

- ClaimContextTests.cc

5.125 secmgr_tests::TestSessionListener Class Reference

Inheritance diagram for secmgr_tests::TestSessionListener:

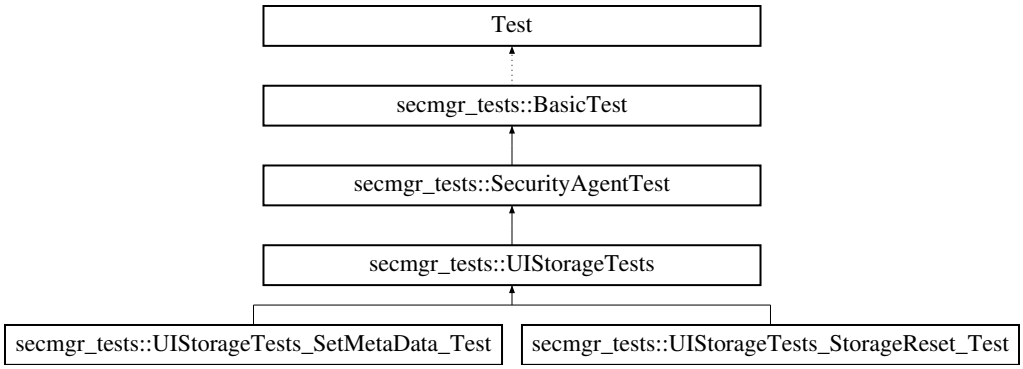


The documentation for this class was generated from the following file:

- TestUtil.h

5.126 secmgr_tests::UIStorageTests Class Reference

Inheritance diagram for secmgr_tests::UIStorageTests:



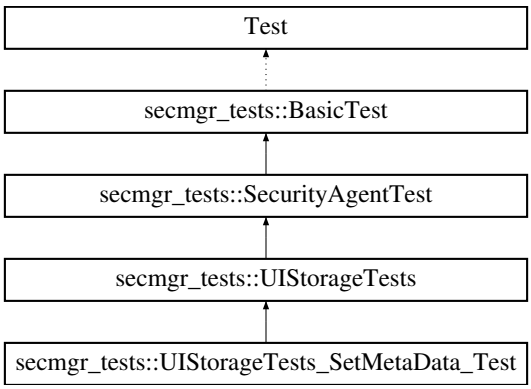
Additional Inherited Members

The documentation for this class was generated from the following file:

- UIStorageTests.cc

5.127 secmgr_tests::UIStorageTests_SetMetaData_Test Class Reference

Inheritance diagram for secmgr_tests::UIStorageTests_SetMetaData_Test:



Additional Inherited Members

5.127.1 Detailed Description

Test Set the user defined name of an application and check whether it can be retrieved.

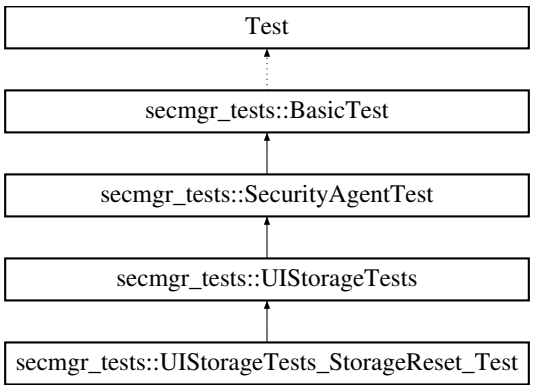
1. Claim the remote application.
2. Set some meta data.
3. Retrieve the application from the security agent.
4. Check whether the retrieved meta data matches the data that was set.

The documentation for this class was generated from the following file:

- UIStorageTests.cc

5.128 secmgr_tests::UIStorageTests_StorageReset_Test Class Reference

Inheritance diagram for secmgr_tests::UIStorageTests_StorageReset_Test:



Additional Inherited Members

5.128.1 Detailed Description

Test Ensure that resetting the database will trigger OnStorageReset.

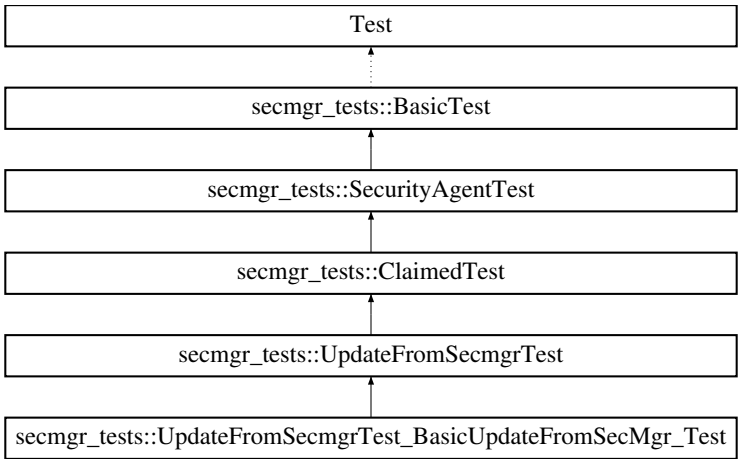
1. Register a storage listener.
2. Reset the storage and make sure that OnStorageReset was called.
3. Unregister the listener.

The documentation for this class was generated from the following file:

- UIStorageTests.cc

5.129 secmgr_tests::UpdateFromSecmgrTest Class Reference

Inheritance diagram for secmgr_tests::UpdateFromSecmgrTest:



Public Types

- enum **UpdateStage** { **UPDATE_STATRED**, **UPDATE_COMPLETED** }

Public Member Functions

- void **TearDown** ()
- shared_ptr< AgentCAStorage > & **GetAgentCAStorage** ()
- bool **WaitForStageUpdates** (UpdateStage stage, size_t numOfApps, size_t until)

Public Attributes

- shared_ptr
  < UpdatesFromSecMgrWrapper > **wrappedCa**
- qcc::Condition **sem**
- qcc::Mutex **lock**

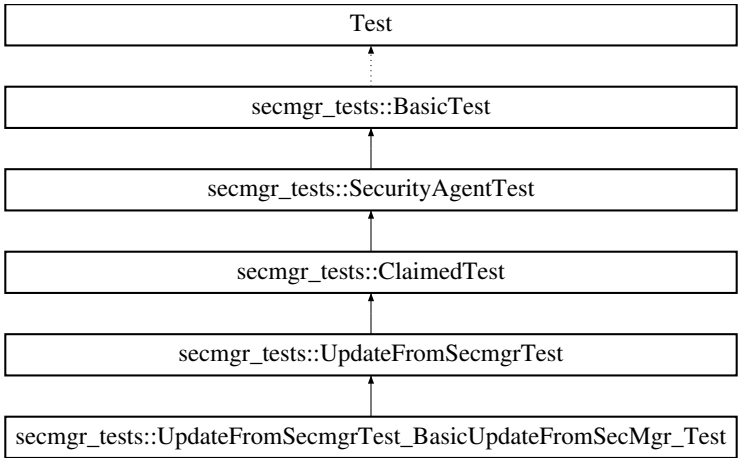
Additional Inherited Members

The documentation for this class was generated from the following file:

- UpdateFromSecMgrTests.cc

5.130 secmgr_tests::UpdateFromSecmgrTest_BasicUpdateFromSecMgr_Test Class Reference

Inheritance diagram for secmgr_tests::UpdateFromSecmgrTest_BasicUpdateFromSecMgr_Test:



Additional Inherited Members

5.130.1 Detailed Description

Test Ensure that an explicit update applications call from the security agent will trigger the correct logic in the application updater.

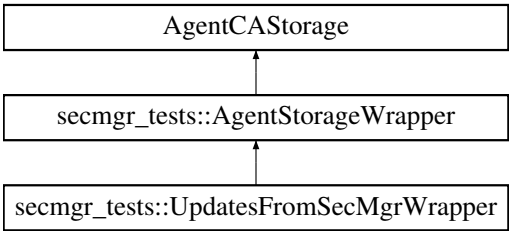
1. Claim an application.
2. Trigger update applications from the security agent for all apps.
3. Ensure that the number of times update started and completed is correct.
4. Trigger update applications from the security agent using a vector containing the claimed app.
5. Ensure that the number of times update started and completed have been incremented.

The documentation for this class was generated from the following file:

- UpdateFromSecMgrTests.cc

5.131 secmgr_tests::UpdatesFromSecMgrWrapper Class Reference

Inheritance diagram for secmgr_tests::UpdatesFromSecMgrWrapper:



Public Member Functions

- **UpdatesFromSecMgrWrapper** (shared_ptr< AgentCAStorage > &_ca)
- QStatus **StartUpdates** (Application &app, uint64_t &updateID)
- QStatus **UpdatesCompleted** (Application &app, uint64_t &updateID)

Public Attributes

- map< Application, size_t > **appsStartedUpdating**
- map< Application, size_t > **appsWithUpdatesCompleted**

Additional Inherited Members

The documentation for this class was generated from the following file:

- UpdateFromSecMgrTests.cc

Chapter 6

File Documentation

6.1 AgentStorageWrapper.h File Reference

Classes

- class secmgr_tests::AgentStorageWrapper
- class secmgr_tests::FailingStorageWrapper

6.2 AJNCaStorageTests.cc File Reference

```
#include <gtest/gtest.h>
#include "SQLStorageConfig.h"
#include "AJNCaStorage.h"
```

Classes

- class secmgr_tests::AJNCaStorageTest
- class secmgr_tests::AJNCaStorageTest_BasicTest_Test

6.3 AJNCaTests.cc File Reference

```
#include <gtest/gtest.h>
#include <stdio.h>
#include <AJNCa.h>
```

Classes

- class secmgr_tests::AJNCaTest_BasicTest_Test

6.4 ApplicationUpdaterTests.cc File Reference

```
#include <gtest/gtest.h>
#include <qcc/GUID.h>
#include <qcc/Util.h>
#include <qcc/Thread.h>
#include <alljoyn/Status.h>
#include <alljoyn/securitymgr/PolicyGenerator.h>
#include "TestUtil.h"
#include "AgentStorageWrapper.h"
```

Classes

- class secmgr_tests::SyncErrorStorageWrapper
- class secmgr_tests::ApplicationUpdaterTests
- class secmgr_tests::ApplicationUpdaterTests_Reset_Test
- class secmgr_tests::ApplicationUpdaterTests_InstallMembership_Test
- class secmgr_tests::ApplicationUpdaterTests_UpdatePolicy_Test
- class secmgr_tests::ApplicationUpdaterTests_ResetPolicy_Test
- class secmgr_tests::ApplicationUpdaterTests_InstallIdentity_Test
- class secmgr_tests::ApplicationUpdaterTests_UpdateAll_Test
- class secmgr_tests::ApplicationUpdaterTests_SyncErReset_Test
- class secmgr_tests::ApplicationUpdaterTests_SyncErPolicy_Test
- class secmgr_tests::ApplicationUpdaterTests_SyncErIdentity_Test
- class secmgr_tests::ApplicationUpdaterTests_SyncErMembership_Test
- class secmgr_tests::ApplicationUpdaterTests_SyncErStorage_Test

6.5 CertChainHandlingTests.cc File Reference

```
#include <gtest/gtest.h>
#include "TestUtil.h"
#include "AgentStorageWrapper.h"
#include "AJNCa.h"
#include <qcc/GUID.h>
#include <alljoyn/Status.h>
#include <alljoyn/securitymgr/CertificateUtil.h>
```

Classes

- class secmgr_tests::CertChainAgentStorageWrapper
- class secmgr_tests::CertChainHandlingTests
- class secmgr_tests::CertChainHandlingTests_ClaimChain_Test
- class secmgr_tests::CertChainHandlingTests_InstallMembershipChain_Test
- class secmgr_tests::CertChainHandlingTests_UpdateIdentityChains_Test
- class secmgr_tests::CertChainHandlingTests_RegisterAgent_Test

Macros

- #define **CHECK_IDENTITY_CHAIN**(c)
- #define **CHECK_MEMBERSHIP_SUMMARIES**()

6.5.1 Macro Definition Documentation

6.5.1.1 #define CHECK_IDENTITY_CHAIN(c)

Value:

```
{ \
        bool failure = true; \
        CheckIdentyCertificateChain((c), failure, __FUNCTION__, __LINE__); \
        if (failure) { return; } \
    }
```

6.5.1.2 #define CHECK_MEMBERSHIP_SUMMARIES( )

Value:

```
{ \
        bool failure = true; \
        CheckMembershipSummaries(failure,  __FUNCTION__, __LINE__); \
        if (failure) { return; } \
    }
```

6.6 CertificateUtilTests.cc File Reference

```
#include "TestUtil.h"
#include <alljoyn/securitymgr/CertificateUtil.h>
#include <qcc/CertificateECC.h>
```

Classes

- class secmgr_tests::CertificateUtilTests
- class secmgr_tests::CertificateUtilTests_ToMembershipCertificate_Test
- class secmgr_tests::CertificateUtilTests_ToIdentityCertificate_Test
- class secmgr_tests::CertificateUtilTests_FailedToMembershipAndIdentityCertificate_Test

6.7 ClaimContextTests.cc File Reference

```
#include "TestUtil.h"
#include <alljoyn/securitymgr/CertificateUtil.h>
#include <qcc/CertificateECC.h>
```

Classes

- class secmgr_tests::TestClaimContext
- class secmgr_tests::ClaimContextTests
- class secmgr_tests::ClaimContextTests_BasicConstructor_Test
- class secmgr_tests::ClaimContextTests_ApproveManifest_Test
- class secmgr_tests::ClaimContextTests_SetClaimType_Test

6.8 ClaimingTests.cc File Reference

```
#include "TestUtil.h"
#include "AgentStorageWrapper.h"
#include <qcc/Thread.h>
```

Classes

- class secmgr_tests::AutoRejector
- class secmgr_tests::RejectAfterAcceptListener
- class secmgr_tests::StopBeforeAcceptListener
- class secmgr_tests::ClaimingTests
- class secmgr_tests::ClaimingTests_SuccessfulClaim_Test
- class secmgr_tests::ClaimingTests_RejectManifest_Test
- class secmgr_tests::ClaimingTests_BasicRobustness_Test
- class secmgr_tests::ClaimingTests_RecoveryFromClaimingFailure_Test
- class secmgr_tests::ClaimingTests_ConcurrentClaimListenerUpdate_Test
- class secmgr_tests::PSKClaimListener
- class secmgr_tests::ClaimingTests_OOBSuccessfulClaiming_Test
- class secmgr_tests::BadPSKClaimListener
- class secmgr_tests::ClaimingTests_OOBFailedClaiming_Test
- class secmgr_tests::BadClaimListener
- class secmgr_tests::ClaimingTests_ClaimListenerErrors_Test
- class secmgr_tests::NestedPSKClaimListener
- class secmgr_tests::ClaimingTests_NestedPSKClaims_Test
- class secmgr_tests::ClaimThread
- class secmgr_tests::ConcurrentPSKClaimListener
- class secmgr_tests::ClaimingTests_ConcurrentPSKClaims_Test
- class secmgr_tests::ClaimingTests_ResetAfterReportClaimFails_Test
- class secmgr_tests::ConcurrentSameClaimListener
- class secmgr_tests::ClaimingTests_ConcurrentClaimOfSameApp_Test

6.9 ConcurrentUpdateTests.cc File Reference

```
#include <gtest/gtest.h>
#include "TestUtil.h"
#include "AgentStorageWrapper.h"
#include <qcc/GUID.h>
#include <qcc/String.h>
#include <qcc/Util.h>
#include <qcc/Thread.h>
#include <alljoyn/Status.h>
#include <alljoyn/securitymgr/PolicyGenerator.h>
```

Classes

- class secmgr_tests::CCAgentStorageWrapper
- class secmgr_tests::ConcurrentUpdateTests
- class secmgr_tests::ConcurrentUpdateTests_ResetAfterPolicy_Test
- class secmgr_tests::ConcurrentUpdateTests_InstallMembershipAfterPolicy_Test
- class secmgr_tests::ConcurrentUpdateTests_UpdatePolicyAfterPolicy_Test
- class secmgr_tests::ConcurrentUpdateTests_UpdateMultiple_Test

Enumerations

- enum **Action** {
  NOTHING, **RESET**, **MEMBERSHIP**, **POLICY**,
  MULTI }

6.10 DiscoveryTests.cc File Reference

```
#include "TestUtil.h"
```

Classes

- class secmgr_tests::DiscoveryTests
- class secmgr_tests::DiscoveryTests_LateJoiningSecurityAgent_Test

6.11 GroupManagementTests.cc File Reference

```
#include <string>
#include "TestUtil.h"
```

Classes

- class secmgr_tests::GroupManagementTests
- class secmgr_tests::GroupManagementTests_GroupManipBasic_Test
- class secmgr_tests::GroupManagementTests_GroupManipManyGroups_Test
- class secmgr_tests::GroupManagementTests_DefaultAuthority_Test
- class secmgr_tests::GroupManagementTests_AuthoritiesCheck_Test
- class secmgr_tests::GroupManagementTests_FailedBasicGroupOperations_Test
- class secmgr_tests::GroupManagementTests_GroupUpdate_Test

6.12 IdentityManagementTests.cc File Reference

```
#include <string>
#include "TestUtil.h"
```

Classes

- class secmgr_tests::IdentityManagementTests
- class secmgr_tests::IdentityManagementTests_IdentityManipBasic_Test
- class secmgr_tests::IdentityManagementTests_IdentityManipManyIdentities_Test
- class secmgr_tests::IdentityManagementTests_FailedBasicIdentityOperations_Test
- class secmgr_tests::IdentityManagementTests_IdentityUpdate_Test

6.13 IdentityTests.cc File Reference

```
#include "TestUtil.h"
```

Classes

- class secmgr_tests::IdentityTests
- class secmgr_tests::IdentityTests_SuccessfulInstallIdentity_Test
- class secmgr_tests::IdentityTests_UpdateIdentityPolicyUpdate_Test

6.14 ManifestTests.cc File Reference

```
#include "TestUtil.h"
#include <qcc/Thread.h>
```

Classes

- class secmgr_tests::ManifestTests
- class secmgr_tests::ManifestTests_UpdateManifest_Test

6.15 ManifestUtilTests.cc File Reference

```
#include <alljoyn/securitymgr/Util.h>
#include "TestUtil.h"
```

Classes

- class secmgr_tests::ManifestUtilTests
- class secmgr_tests::ManifestUtilTests_ManifestConstruction_Test
- class secmgr_tests::ManifestUtilTests_ExtendedPermissionPolicyDigest_Test
- class secmgr_tests::ManifestUtilTests_ManifestIllegalArgs_Test
- class secmgr_tests::ManifestUtilTests_UtilIllegalArgs_Test
- class secmgr_tests::ManifestUtilTests_Difference_Test

6.16 MembershipTests.cc File Reference

```
#include <qcc/Thread.h>
#include "TestUtil.h"
```

Classes

- class secmgr_tests::MembershipTests
- class secmgr_tests::MembershipTests_SuccessfulInstallMembership_Test
- class secmgr_tests::MembershipTests_InstallRemoveMembershipPolicyUpdate_Test

6.17 MultiAgentAppTests.cc File Reference

```
#include "TestUtil.h"
```

Classes

- class secmgr_tests::MultiAgentAppTests
- class secmgr_tests::MultiAgentAppTests_DISABLED_SuccessfulClaimAndUnclaimManyApps_Test
- class secmgr_tests::MultiAgentAppTests_DISABLED_SuccessfulUpdateManyApps_Test
- class secmgr_tests::MultiAgentAppTests_DISABLED_SuccessfulRestartSecAgentWithManyApps_Test

6.18 PolicyTests.cc File Reference

```
#include <alljoyn/securitymgr/PolicyGenerator.h>
#include "TestUtil.h"
```

Classes

- class secmgr_tests::PolicyTests
- class secmgr_tests::PolicyTests_SuccessfulInstallPolicyAndUpdatePolicy_Test
- class secmgr_tests::PolicyTests_SuccessfulResetPolicy_Test
- class secmgr_tests::PolicyTests_PermissionDenied_Test

6.19 PolicyUtilTests.cc File Reference

```
#include <gtest/gtest.h>
#include <alljoyn/securitymgr/PolicyUtil.h>
#include "TestUtil.h"
```

Classes

- class secmgr_tests::PolicyUtilTest
- class secmgr_tests::PolicyUtilTest_NormalizePolicyMembers_Test
- class secmgr_tests::PolicyUtilTest_NormalizePolicyRules_Test
- class secmgr_tests::PolicyUtilTest_NormalizePolicy_Test

6.20 ResetTests.cc File Reference

```
#include "TestUtil.h"
#include "AgentStorageWrapper.h"
```

Classes

- class secmgr_tests::ResetTests
- class secmgr_tests::ResetTests_SuccessfulReset_Test
- class secmgr_tests::ResetTests_RecoveryFromResetFailure_Test
- class secmgr_tests::ResetTests_RecoveryFromResetSuccess_Test
- class secmgr_tests::ResetTests_RecoveryFromRemoteReset_Test

6.21 RestartAgentTests.cc File Reference

```
#include "TestUtil.h"
```

Classes

- class secmgr_tests::RestartAgentTests
- class secmgr_tests::RestartAgentTests_SuccessfulAgentRestart_Test

6.22 SecMgrCoreTest.cc File Reference

```
#include <gtest/gtest.h>
#include <alljoyn/Init.h>
```

Functions

- int CDECL_CALL **main** (int argc, char **argv)

6.23 SecurityAgentFactoryTests.cc File Reference

```
#include "TestUtil.h"
```

Classes

- class secmgr_tests::SecurityAgentFactoryTests
- class secmgr_tests::SecurityAgentFactoryTests_Basic_Test

6.24 TestApplication.cc File Reference

```
#include <string.h>
#include <alljoyn/AuthListener.h>
#include "TestApplication.h"
#include "AJNCa.h"
```

6.25 TestApplication.h File Reference

```
#include <string>
#include <memory>
#include <alljoyn/BusAttachment.h>
#include <alljoyn/securitymgr/Manifest.h>
#include <qcc/GUID.h>
```

Classes

- class secmgr_tests::TestAppAuthListener
- class secmgr_tests::TestApplication

6.26 TestUtil.cc File Reference

```
#include "TestUtil.h"
#include <stdlib.h>
#include <qcc/Thread.h>
#include <qcc/Util.h>
#include <qcc/Environ.h>
#include <alljoyn/securitymgr/Util.h>
#include "SQLStorageConfig.h"
```

Functions

- string **secmgr_tests::ToString** (const SyncError *er)
- string **secmgr_tests::ManifestUpdateToString** (const ManifestUpdate *update)

6.27 TestUtil.h File Reference

```
#include <gtest/gtest.h>
#include <memory>
#include <qcc/String.h>
#include <qcc/Mutex.h>
#include <qcc/Condition.h>
#include <ProxyObjectManager.h>
#include <alljoyn/securitymgr/Application.h>
#include <alljoyn/securitymgr/SecurityAgentFactory.h>
#include <alljoyn/securitymgr/SyncError.h>
#include <alljoyn/securitymgr/storage/StorageFactory.h>
#include <alljoyn/securitymgr/PolicyGenerator.h>
#include "TestApplication.h"
```

Classes

- class secmgr_tests::TestSessionListener
- class secmgr_tests::TestAboutListener
- class secmgr_tests::TestApplicationListener
- class secmgr_tests::AutoAccepter
- class secmgr_tests::BasicTest
- class secmgr_tests::SecurityAgentTest
- class secmgr_tests::ClaimedTest

Macros

- #define **TEST_STORAGE_NAME** "test"

6.28 UIStorageTests.cc File Reference

```
#include "TestUtil.h"
```

Classes

- class secmgr_tests::StorageListenerReset
- class secmgr_tests::UIStorageTests
- class secmgr_tests::UIStorageTests_SetMetaData_Test
- class secmgr_tests::UIStorageTests_StorageReset_Test

6.29 UpdateFromSecMgrTests.cc File Reference

```
#include <gtest/gtest.h>
#include "TestUtil.h"
#include "AgentStorageWrapper.h"
#include <qcc/Util.h>
#include <qcc/Thread.h>
#include <alljoyn/Status.h>
```

Classes

- class secmgr_tests::UpdatesFromSecMgrWrapper
- class secmgr_tests::UpdateFromSecmgrTest
- class secmgr_tests::UpdateFromSecmgrTest_BasicUpdateFromSecMgr_Test

Index