Government Smart Card Interoperability Specification v2.1 (NISTIR 6887 – 2003 Edition) Basic Services Interface C Binding

Conformance Test Instantiation, Verification, and Reporting Scenarios

FINAL DRAFT

Alan Goldfine

January 21, 2005

1. gscBsiUtilAcquireContext()

Starting State for Each Test:

1. A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard0100.

2. There is no authenticated session established with any container on the connected card.

3. There exists a container on the connected card for which
   - readTagListACR is PIN Protected
   - the value of the PIN is _PIN
   - the container is represented by AID value ==
     - _goodGSCAID1, of length _goodGSCAID1Len (GSC), or
     - _goodCACAID1, of length _goodCACAID1Len (CAC).

4. There does not exist a container on the card with AID value ==
   - _badGSCAID, of length _badGSCAIDLen (GSC), or
   - _badCACAID, of length _badCACAIDLen (CAC).

5. There exists a BSIAuthenticator array strctAuthenticator0100 with one element, the structure BSIAuthenticator0100. This structure has fields
   - unAccessMethodType == BSI_AM_PIN
   - unkeyIDOrReference == _keyIDOrReference1
   - uszAuthValue == _goodAuthValue1
   - unAuthValueLen == _goodAuthValue1Len.

6. There exists an unsigned long * punNbTags0100 == 0.

Test for Assertion 1.1

The function is tested using valid parameters.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiUtilAcquireContext().

2. (Pre) Print "Testing of Assertion 1.1".

3. Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard0100
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - strctAuthenticator == strctAuthenticator0100
   - unAuthNb == 1.

Verification Goal:

To verify the Expected Results:

1. The call returns
   - the return code BSI_OK or the return code BSI_TERMINAL_AUTH.

2. An authenticated session is established with the target container.

Perform this verification by issuing a call to gscBsiGcReadTagList().

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiUtilAcquireContext() call returns the code BSI_OK or BSI_TERMINAL_AUTH, then verify that an authenticated session has indeed been established with the target container.

   Make a gscBsiGcReadTagList() call to the SPS, using
   - hCard == hCard0100
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - tagArray == NULL
   - punNbTags == punNbTags0100.

   Case 1.1: If the gscBsiGcReadTagList() call returns the code BSI_OK, then print
   "gscBsiUtilAcquireContext() called with valid parameters has been verified because a subsequent call to gscBsiGcReadTagList() was successful, indicating that an authenticated session had been established.
   Status: Test 1.1 Passed."

   Case 1.2: If the gscBsiGcReadTagList() call does not return the code BSI_OK, then print
   "gscBsiUtilAcquireContext() called with valid parameters has not been verified because a subsequent call to gscBsiGcReadTagList() was unsuccessful, indicating that an authenticated session had not been established.
   Status: Test 1.1 Failed."

   Case 2: If the gscBsiUtilAcquireContext() call does not return the code BSI_OK, then print
   "gscBsiUtilAcquireContext() called with valid parameters returned an incorrect code.
   Status: Test 1.1 Failed."

Test for Assertion 1.2

The function is tested using a bad handle.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiUtilAcquireContext().

2. (Pre) Print "Testing of Assertion 1.2".

3. Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard /= hCard0100
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - strctAuthenticator == strctAuthenticator0100
   - unAuthNb == 1.

Verification Goal:

To verify the Expected Results:

1. The call returns
   - the return code BSI_BAD_HANDLE (a BSIException is thrown, with BSIException.getErrorCode returning BSI_BAD_HANDLE).

2. An authenticated session is not established with the target container.

Perform this verification by issuing a call to gscBsiGcReadTagList().

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiUtilAcquireContext() call returns the code BSI_BAD_HANDLE, then:

   Perform Test for Assertion 9.1.2.1, using
   - hCard == hCard0100

   Verify that an authenticated session has not been established with the target container:

   Make a gscBsiGcReadTagList() call to the SPS, using
   - hCard == hCard0100
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - tagArray == NULL
   - punNbTags == punNbTags0100.

   Case 1.1: If the gscBsiGcReadTagList() call returns the code BSI_ACCESS_DENIED, then print
   "gscBsiUtilAcquireContext() called with a bad handle has been verified because a subsequent call to gscBsiGcReadTagList() resulted in a denial of access, indicating that an authenticated session had not been established.
   Status: Test 1.2 Passed."

   Case 1.2: If the gscBsiGcReadTagList() call does not return the code BSI_ACCESS_DENIED, then print
   "gscBsiUtilAcquireContext() called with a bad handle has not been verified because a subsequent call to gscBsiGcReadTagList() did not result in a denial of access, indicating that an authenticated session had been established.
   Status: Test 1.2 Failed."

   Case 2: If the gscBsiUtilAcquireContext() call does not return the code BSI_BAD_HANDLE, then print
   "gscBsiUtilAcquireContext() called with a bad handle returned an incorrect code.
   Status: Test 1.2 Failed."

Test for Assertion 1.3

The function is tested using a bad AID value.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiUtilAcquireContext().

2. (Pre) Print "Testing of Assertion 1.2".

3. Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard0100
   - uszAID == _badGSCAID (GSC) or _badCACAID (CAC)
   - unAIDLen == _badGSCAIDLen (GSC) or _badCACAIDLen (CAC)
   - strctAuthenticator == strctAuthenticator0100
   - unAuthNb == 1.

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_BAD_AID.

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiUtilAcquireContext() call returns the code BSI_BAD_AID, then:

   Perform Test for Assertion 9.1.3.1 using
   - hCard == hCard0100.

   Print
   "gscBsiUtilAcquireContext() called with a bad AID value has been verified.
   Status: Test 1.3 Passed."

   Case 2: If the gscBsiUtilAcquireContext() call does not return the code BSI_BAD_AID, then print
   "gscBsiUtilAcquireContext() called with a bad AID value returned an incorrect code.
   Status: Test 1.3 Failed."

Test for Assertion 1.4

The function is tested using a bad AID length.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiUtilAcquireContext().

2. (Pre) Define the unsigned long unAIDLen0104 == 0.

3. (Pre) Print "Testing of Assertion 1.4".

4. Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard0100
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _badAIDLen
   - strctAuthenticator == strctAuthenticator0100
   - unAuthNb == 1.

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_BAD_PARAM.

2. An authenticated session is not established with the target container.

Perform this verification by issuing a call to gscBsiGcReadTagList().

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiUtilAcquireContext() call returns the code BSI_BAD_PARAM, then:

   Perform Test for Assertion 9.1.4.1 using
   - hCard == hCard0100.

   Verify that an authenticated session has not been established with the target container:

   Make a gscBsiGcReadTagList() call to the SPS, using
   - hCard == hCard0100
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - tagArray == NULL
   - punNbTags == punNbTags0100.

   Case 1.1: If the gscBsiGcReadTagList() call returns the code BSI_ACCESS_DENIED, then print
   "gscBsiUtilAcquireContext() called with a bad AID length has been verified because a subsequent call to gscBsiGcReadTagList() resulted in a denial of access, indicating that an authenticated session had not been established.
   Status: Test 1.4 Passed."

   Case 1.2: If the gscBsiGcReadTagList() call does not return the code BSI_ACCESS_DENIED, then print
   "gscBsiUtilAcquireContext() called with a bad AID length has not been verified because a subsequent call to gscBsiGcReadTagList() did not result in a denial of access, indicating that an authenticated session had been established.
   Status: Test 1.4 Failed."

   Case 2: If the gscBsiUtilAcquireContext() call does not return the code BSI_BAD_PARAM, then print
   "gscBsiUtilAcquireContext() called with a bad AID length returned an incorrect code.
   Status: Test 1.4 Failed."

Test for Assertion 1.5

The function is tested using an authentication method that is not available on the card.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiUtilAcquireContext().

2. (Pre) Allocate the unsigned long* punChallengeLen0105 == 0.

3. Make a gscBsiGetChallenge() call to the SPS, using
   - hCard == hCard0105
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - uszChallenge == NULL
   - punChallengeLen == punChallengeLen0105.

   Case 1: If the gscBsiGetChallenge() call returns the code BSI_OK, then continue with 4.

   Case 2: If the gscBsiGetChallenge() call does not return the code BSI_OK, then print
   "gscBsiGetChallenge() cannot be completed.
   Status: Assertion 1.5 of gscBsiUtilAcquireContext() cannot be tested."
   End Test for Assertion 1.5.

4. Allocate the unsigned string uszChallenge0105 with length == punChallengeLen0105*.

5. Make a gscBsiGetChallenge() call to the SPS, using
   - hCard == hCard0105
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - uszChallenge == uszChallenge0105
   - punChallengeLen == punChallengeLen0105.

6. (Pre) Construct a BSIAuthenticator array strctAuthenticator0105 with the element BSIAuthenticator0105. BSIAuthenticator0105 has fields
   - unAccessMethodType == BSI_AM_XAUTH
   - unkeyIDOrReference == _keyIDOrReference1
   - uszAuthValue == uszChallenge0105
   - unAuthValueLen == punChallengeLen0105.

Note: The gscBsiGetChallenge() call was issued because the call itself may be required by an implementation prior to an gscBsiUtilAcquireContext() call; the actual content of uszChallenge0105 in BSIAuthenticator0105 is unimportant.

7. (Pre) Print "Testing of Assertion 1.5".

8. Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard0100
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - strctAuthenticator == strctAuthenticator0105
   - unAuthNb == 1.

Verification Goal:

To verify the Expected Results:

1. The call returns
   - the return code BSI_ACR_NOT_AVAILABLE.

2. An authenticated session is not established with the target container.

Perform this verification by issuing a call to gscBsiGcReadTagList().

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiUtilAcquireContext() call returns the code BSI_ACR_NOT_AVAILABLE, then:

   Perform Test for Assertion 9.1.5.1 using
   - hCard == hCard0100.

   Verify that an authenticated session has not been established with the target container:

   Make a gscBsiGcReadTagList() call to the SPS, using
   - hCard == hCard0100
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - tagArray == NULL
   - punNbTags == punNbTags0100.

   Case 1.1: If the gscBsiGcReadTagList() call returns the code BSI_ACCESS_DENIED, then print
   "gscBsiUtilAcquireContext() called with an authentication method that is not available on the card has been verified because a subsequent call to gscBsiGcReadTagList() resulted in a denial of access, indicating that an authenticated session had not been established.
   Status: Test 1.5 Passed."

   Case 1.2: If the gscBsiGcReadTagList() call does not return the code BSI_ACCESS_DENIED, then print
   "gscBsiUtilAcquireContext() called with an authentication method that is not available on the card has not been verified because a subsequent call to gscBsiGcReadTagList() did not result in a denial of access, indicating that an authenticated session had been established.
   Status: Test 1.5 Failed."

   Case 2: If the gscBsiUtilAcquireContext() call does not return the code BSI_ACR_NOT_AVAILABLE, then print
   "gscBsiUtilAcquireContext() called with an authentication method that is not available on the card returned an incorrect code.
   Status: Test 1.5 Failed."

Test for Assertion 1.6

The function is tested using a bad authenticator.

Verification Goal:
To verify the Expected Results

1. The call returns
   - the return code BSI_BAD_AUTH.

2. An authenticated session is not established with the target container.

Perform this verification by issuing a call to gscBsiGcReadTagList().

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiUtilAcquireContext().

2. (Pre) Construct a BSIAuthenticator array strctAuthenticator0106 with the element BSIAuthenticator0106. BSIAuthenticator0106 has fields
   - unAccessMethodType == BSI_AM_PIN
   - unkeyIDOrReference == _keyIDOrReference1
   - uszAuthValue == _badAuthValue
   - unAuthValueLen == _badAuthValueLen.

3. (Pre) Print "Testing of Assertion 1.6".

4. Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard0100
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - strctAuthenticator == strctAuthenticator0106
   - unAuthNb == 1.

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_BAD_AUTH.

2. An authenticated session is not established with the target container.

Perform this verification by issuing a call to gscBsiGcReadTagList().

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiUtilAcquireContext() call returns the code BSI_BAD_AUTH, then:

   Perform Test for Assertion 9.1.6.1 using
   - hCard == hCard0100.

   Verify that an authenticated session has not been established with the target container:

   Make a gscBsiGcReadTagList() call to the SPS, using
   - hCard == hCard0100
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)

- tagArray == NULL
- punNbTags == punNbTags0100.

Case 1.1: If the gscBsiGcReadTagList() call returns the code BSI_ACCESS_DENIED, then print
"gscBsiUtilAcquireContext() called with a bad authenticator has been verified because a subsequent call to gscBsiGcReadTagList() resulted in a denial of access, indicating that an authenticated session had not been established.
Status: Test 1.6 Passed."

Case 1.2: If the gscBsiGcReadTagList() call does not return the code BSI_ACCESS_DENIED, then print
"gscBsiUtilAcquireContext() called with a bad authenticator has not been verified because a subsequent call to gscBsiGcReadTagList() did not result in a denial of access, indicating that an authenticated session had been established.
Status: Test 1.6 Failed."

Case 2: If the gscBsiUtilAcquireContext() call does not return the code BSI_BAD_AUTH, then print
"gscBsiUtilAcquireContext() called with a bad authenticator returned an incorrect code.
Status: Test 1.6 Failed."

Test for Assertion 1.7

The function is tested with a card that has been removed.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiUtilAcquireContext().

2. (Pre) Remove the connected card from the reader.

3. (Pre) Print "Testing of Assertion 1.7".

4. Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard0100
   - uszAID == AID0100
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - strctAuthenticator == strctAuthenticator0100
   - unAuthNb == 1.

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_CARD_REMOVED.

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiUtilAcquireContext() call returns the code BSI_CARD_REMOVED, then print
   "gscBsiUtilAcquireContext() called with the connected card removed has been verified.

Status: Test 1.7 Passed."

Case 2: If the gscBsiUtilAcquireContext() call does not return the code BSI_CARD_REMOVED, then print
"gscBsiUtilAcquireContext() called with the connected card removed returned an incorrect code.
Status: Test 1.7 Failed."

Test for Assertion 1.8

The function is tested with another application having established a transaction lock.

Until we encounter implementations that allow multiple simultaneous applications, I don't think we need to worry about this assertion.

Test for Assertion 1.9

The function is tested with a blocked PIN.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiUtilAcquireContext().

2. (Pre) Construct a BSIAuthenticator array strctAuthenticator0109 with one element, the structure BSIAuthenticator0109. BSIAuthenticator0109 has fields
   - accessMethodType0109 == BSI_AM_PIN
   - keyIDOrReference0109 == _keyIDOrReference2
   - authValue0109 == _goodAuthValue2
   - authValue0109Len == _goodAuthValue2Len.

3. (Pre) Make N gscBsiUtilAcquireContext() calls to the SPS, using
   - hCard == hCard0100
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - strctAuthenticator == strctAuthenticator0109
   - unAuthNb == 1

   where N is the maximum number of incorrect PIN tries allowed by the implementation.

4. (Pre) Print "Testing of Assertion 1.8".

5. Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard0100
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - strctAuthenticator == strctAuthenticator0100
   - unAuthNb == 1.

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_PIN_BLOCKED.

2. An authenticated session is not established with the target container.

Perform this verification by issuing a call to gscBsiGcReadTagList().

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiUtilAcquireContext() call returns the code BSI_PIN_BLOCKED, then:

   Perform Test for Assertion 9.1.9.1 using
   - hCard == hCard0100.

   Verify that an authenticated session has not been established with the target container:

   Make a gscBsiGcReadTagList() call to the SPS, using
   - hCard == hCard0100
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - tagArray == NULL
   - punNbTags == punNbTags0100.

   Case 1.1: If the gscBsiGcReadTagList() call returns the code BSI_ACCESS_DENIED, then print
   "gscBsiUtilAcquireContext() called with a blocked PIN has been verified because a subsequent call to gscBsiGcReadTagList() resulted in a denial of access, indicating that an authenticated session had not been established.
   Status: Test 1.9 Passed."

   Case 1.2: If the gscBsiGcReadTagList() call does not return the code BSI_ACCESS_DENIED, then print
   "gscBsiUtilAcquireContext() called with a blocked PIN has not been verified because a subsequent call to gscBsiGcReadTagList() did not result in a denial of access, indicating that an authenticated session had been established.
   Status: Test 1.9 Failed."

   Case 2: If the gscBsiUtilAcquireContext() call does not return the code BSI_PIN_BLOCKED, then print
   "gscBsiUtilAcquireContext() called with a blocked PIN returned an incorrect code.
   Status: Test 1.9 Failed."

2. gscBsiUtilConnect()

Starting State for Each Test:

1. There exists a card reader, whose name is represented by the string uszReaderName0200, available to the candidate implementation.

2. The length of uszReaderName0200 is unReaderNameLen0200.

3. There exists an unsigned long * hCard0200.

Test for Assertion 2.1

The function is tested using valid parameters, with a good card inserted into a specified reader.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiUtilConnect().

2. (Pre) Insert a card that claims conformance to the GSC-IS into the reader uszReaderName0200.

3. (Pre) Print "Testing of Assertion 2.1".

4. Make an gscBsiUtilConnect() call to the SPS, with
   - uszReaderName == uszReaderName0200
   - unReaderNameLen == unReaderNameLen0200
   - hCard == hCard0200.

Verification Goal:

To verify the Expected Results:

1. The call returns
   - the return code BSI_OK
   - hCard0200 == a valid handle.

2. The card is connected, with handle hCard0200, to the reader uszReaderName0200.

Perform this verification by issuing a call to gscBsiUtilGetCardStatus().

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiUtilConnect() call returns the code BSI_OK, then verify that the card is indeed connected:

   Make a gscBsiUtilGetCardStatus() call to the SPS, using
   - hCard == hCard0200.

   Case 1.1: If the gscBsiUtilGetCardStatus() call returns the code BSI_OK, then print
   "gscBsiUtilConnect() called using valid parameters with a good card inserted into a specified reader has been verified

because a subsequent call to gscBsiUtilGetCardStatus() was successful, indicating that the card had been connected.
Status: Test 2.1 Passed."

Case 1.2: If the gscBsiUtilGetCardStatus() call does not return the code BSI_OK, then print
"gscBsiUtilConnect() called using valid parameters with a good card inserted into a specified reader has not been verified because a subsequent call to gscBsiUtilGetCardStatus() was unsuccessful, indicating that the card had not been connected.
Status: Test 2.1 Failed."

Case 2: If the gscBsiUtilConnect() call does not return the code BSI_OK, then print
"gscBsiUtilConnect() called using valid parameters with a good card inserted into a specified reader returned an incorrect code.
Status: Test 2.1 Failed."

Test for Assertion 2.2

The function is tested using valid parameters, with a good card inserted into a non-specified reader.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiUtilConnect().

2. (Pre) Insert a card that claims conformance to the GSC-IS into the reader uszReaderName0200.

3. (Pre) Print "Testing of Assertion 2.2".

4. Make an gscBsiUtilConnect() call to the SPS, with
   - uszReaderName == ""
   - unReaderNameLen == 0
   - hCard == hCard0200.

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_OK
   - hCard0200 == a valid handle.

2. The card is connected, with handle hCard0200, to the first available reader.

Perform this verification by issuing a call to gscBsiUtilGetCardStatus().

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiUtilConnect() call returns the code BSI_OK, then verify that the card is indeed connected:

   Make a gscBsiUtilGetCardStatus() call to the SPS, using

- hCard == hCard0200.

Case 1.1: If the gscBsiUtilGetCardStatus() call returns the code BSI_OK, then print
"gscBsiUtilConnect() called using valid parameters with a good card inserted into a non-specified reader has been verified because a subsequent call to gscBsiUtilGetCardStatus() was successful, indicating that the card had been connected.
Status: Test 2.2 Passed."

Case 1.2: If the gscBsiUtilGetCardStatus() call does not return the code BSI_OK, then print
"gscBsiUtilConnect() called using valid parameters with a good card inserted into a non-specified reader has not been verified because a subsequent call to gscBsiUtilGetCardStatus() was unsuccessful, indicating that the card had not been connected.
Status: Test 2.2 Failed."

Case 2: If the gscBsiUtilConnect() call does not return the code BSI_OK, then print
"gscBsiUtilConnect() called using valid parameters with a good card inserted into a non-specified reader returned an incorrect code.
Status: Test 2.2 Failed."

Test for Assertion 2.3

The function is tested using a bad reader name length.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiUtilConnect().

2. (Pre) Insert a card that claims conformance to the GSC-IS into the reader uszReaderName0200.

3. (Pre) Print "Testing of Assertion 2.3".

4. Make an gscBsiUtilConnect() call to the SPS, with
   - uszReaderName == uszReaderName0200
   - unReaderNameLen == 0
   - hCard == hCard0200.

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_BAD_PARAM.

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiUtilConnect() call returns the code BSI_BAD_PARAM, then print
   "gscBsiUtilConnect() called with a bad reader name length has been verified.
   Status: Test 2.3 Passed."

Case 2: If the gscBsiUtilConnect() call does not return the code BSI_BAD_PARAM, then print
"gscBsiUtilConnect() called with a bad reader name length returned an incorrect code.
Status: Test 2.3 Failed."

Test for Assertion 2.4

The function is tested using a bad reader name.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiUtilConnect().

2. (Pre) Insert a card that claims conformance to the GSC-IS into the reader uszReaderName0200.

3. (Pre) Print "Testing of Assertion 2.4".

4. Make an gscBsiUtilConnect() call to the SPS, with
   - uszReaderName == _badReaderName
   - unReaderNameLen == _badReaderNameLen
   - hCard == hCard0200.

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_UNKNOWN_READER.

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiUtilConnect() call returns the code BSI_UNKNOWN_READER, then print
   "gscBsiUtilConnect() called with a bad reader name has been verified.
   Status: Test 2.4 Passed."

   Case 2: If the gscBsiUtilConnect() call does not return the code BSI_UNKNOWN_READER, then print
   "gscBsiUtilConnect() called with a bad reader name returned an incorrect code.
   Status: Test 2.4 Failed."

Test for Assertion 2.5

The function is tested with no card in the reader.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiUtilConnect().

2. (Pre) Manually ensure that there is no valid card in the particular reader represented by the String uszReaderName0200.

3. (Pre) Print "Testing of Assertion 2.5".

4. Make an gscBsiUtilConnect() call to the SPS, with
   - uszReaderName == uszReaderName0200
   - unReaderNameLen == unReaderNameLen0200
   - hCard == hCard0200.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_CARD_ABSENT.

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiUtilConnect() call returns the code BSI_CARD_ABSENT, then print
   "gscBsiUtilConnect() called with no card in the reader has been verified.
   Status: Test 2.5 Passed."

   Case 2: If the gscBsiUtilConnect() call does not return the code BSI_CARD_ABSENT, then print
   "gscBsiUtilConnect() called with no card in the reader returned an incorrect code.
   Status: Test 2.5 Failed."

Test for Assertion 2.6

The function is tested using a bad inserted card.

Instantiation Scenario:
1. (Pre) Construct the Starting State for the testing of gscBsiUtilConnect().

2. (Pre) Insert an invalid card into the reader uszReaderName0200.

3. (Pre) Print "Testing of Assertion 2.6".

4. Make an gscBsiUtilConnect() call to the SPS, with
   - uszReaderName == uszReaderName0200
   - unReaderNameLen == unReaderNameLen0200
   - hCard == hCard0200.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_CARD_ABSENT or BSI_UNKNOWN_ERROR.

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiUtilConnect() call returns the code BSI_CARD_ABSENT or the code BSI_UNKNOWN_ERROR, then print
   "gscBsiUtilConnect() called with a bad inserted card has been verified.
   Status: Test 2.6 Passed."

   Case 2: If the gscBsiUtilConnect() call does not return the code BSI_CARD_ABSENT or the code BSI_UNKNOWN_ERROR, then print

"gscBsiUtilConnect() called with a bad inserted card returned an incorrect code.
Status: Test 2.6 Failed."

3. gscBsiUtilDisconnect()

Starting State for Each Test:

1. A card that claims conformance to the GSC-IS is in a particular reader available to the candidate implementation.

2. The card is not connected.

3. The name of the reader is represented by the string uszReaderName0300, of length unReaderNameLen0300.

4. There exists an int hCard0300.

5. Make a gscBsiUtilConnect() call to the SPS, with
   - uszReaderName == uszReaderName0300
   - unReaderNameLen == unReaderNameLen0300
   - hCard == hCard0300.

Test for Assertion 3.1

The function is tested using valid parameters.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiUtilDisconnect().

2. (Pre) Print "Testing of Assertion 3.1".

3. Make a gscBsiUtilDisconnect() call to the SPS, with
   - hCard == hCard0300.

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_OK.

2. The card is disconnected.

Perform this verification by issuing a call to GscBsiUtilGetCardStatus().

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiUtilDisconnect() call returns the code BSI_OK, then verify that the card is indeed disconnected:

   Make a gscBsiUtilGetCardStatus() call to the SPS, using
   - hCard == hCard0300.

   Case 1.1: If the gscBsiUtilGetCardStatus() call does not return the code BSI_OK, then print
   "gscBsiUtilDisconnect() called with valid parameters has been verified because a subsequent call to gscBsiUtilGetCardStatus() was not successful, indicating that the card was no longer connected.
   Status: Test 3.1 Passed."

Case 1.2: If the gscBsiUtilGetCardStatus() call returns the code BSI_OK, then print
"gscBsiUtilDisconnect() called with valid parameters has not been verified because a subsequent call to gscBsiUtilGetCardStatus() was successful, indicating that the card was still connected.
Status: Test 3.1 Failed."

Case 2: If the gscBsiUtilDisconnect() call does not return the code BSI_OK, then print
"gscBsiUtilDisconnect() called with valid parameters returned an incorrect code.
Status: Test 3.1 Failed."

Test for Assertion 3.2

The function is tested using a bad handle.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiUtilDisconnect().

2. (Pre) Print "Testing of Assertion 3.2".

3. Make a gscBsiUtilDisconnect() call to the SPS, using
   - hCard /= hCard0300.

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_BAD_HANDLE.

2. The card is still connected.

Perform this verification by issuing a call to GscBsiUtilGetCardStatus().

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiUtilDisconnect() call returns the code BSI_BAD_HANDLE, then:

   Perform Test for Assertion 9.3.2.1 using
   - hCard == hCard0300.

   Verify that the card is still connected:

   Make a gscBsiUtilGetCardStatus() call to the SPS, using
   - hCard == hCard0300.

   Case 1.1: If the gscBsiUtilGetCardStatus() call returns the code BSI_OK, then print
   "gscBsiUtilDisconnect() called with a bad handle has been verified because a subsequent call to gscBsiUtilGetCardStatus() was successful, indicating that the card was still connected.

Status: Test 3.2 Passed."

Case 1.2: If the gscBsiUtilGetCardStatus() call does not return the code BSI_OK, then print
"gscBsiUtilDisconnect() called with a bad handle has not been verified because a subsequent call to gscBsiUtilGetCardStatus() was unsuccessful, indicating that the card was no longer connected.
Status: Test 3.2 Failed."

Case 2: If the gscBsiUtilDisconnect() call does not return the code BSI_BAD_HANDLE, then print
"gscBsiUtilDisconnect() called with a bad handle returned an incorrect code.
Status: Test 3.2 Failed."

Test for Assertion 3.3

The function is tested with a card that has been removed.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiUtilDisconnect().

2. (Pre) Print "Testing of Assertion 3.3".

3. Remove the connected card from the reader.

4. Make a gscBsiUtilDisconnect() call to the SPS, with
   - hCard == hCard0300.

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_CARD_REMOVED.

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiUtilDisconnect() call returns the code BSI_CARD_REMOVED, then print
   "gscBsiUtilDisconnect() called with the connected card removed has been verified.
   Status: Test 3.3 Passed."

   Case 2: If the gscBsiUtilDisconnect() call does not return the code BSI_CARD_REMOVED, then print
   "gscBsiUtilDisconnect() called with the connected card removed returned an incorrect code.
   Status: Test 3.3 Failed."

4. gscBsiUtilBeginTransaction()

Starting State for Each Test:

1. A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard0400.

Test for Assertion 4.1

The function is tested as a blocking transaction call, with no existing transaction lock, using valid parameters.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiUtilBeginTransaction().

2. (Pre) Print "Testing of Assertion 4.1".

3. (Pre) Ensure that there is no existing transaction lock:

   Make a gscBsiUtilEndTransaction() call to the SPS, with
   - hCard == hCard0400.

   Case 1: If the gscBsiUtilEndTransaction() call returns either of the codes BSI_OK or BSI_NOT_TRANSACTED, then continue with 4.

   Case 2: If the gscBsiUtilEndTransaction() call does not return either of the codes BSI_OK or BSI_NOT_TRANSACTED, then print "It cannot be assured that there is no existing transaction lock. gscBsiUtilBeginTransaction() cannot be tested."
   End Test for Assertion 4.1.

4. Make a gscBsiUtilBeginTransaction() call to the SPS, with
   - hCard == hCard0400
   - blType == TRUE.

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_OK or the return code BSI_NO_SPSSERVICE.

2. If the return code is BSI_OK, then a transaction is established with the smart card.

Perform this verification by issuing a call to gscBsiUtilEndTransaction().

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiUtilBeginTransaction() call returns the code BSI_OK, then verify that a transaction has indeed been started:

   Make a gscBsiUtilEndTransaction() call to the SPS, with
   - hCard == hCard0400.

Case 1.1: If the gscBsiUtilEndTransaction() call returns the code BSI_OK, then print
"gscBsiUtilBeginTransaction() tested as a blocking transaction call, with no existing transaction lock, using valid parameters, has been verified because a subsequent call to gscBsiUtilEndTransaction() was successful, indicating that a transaction had been started.
Status: Test 4.1 Passed."

Case 1.2: If the gscBsiUtilEndTransaction() call does not return the code BSI_OK, then print
"gscBsiUtilBeginTransaction() tested as a blocking transaction call, with no existing transaction lock, using valid parameters, has not been verified because a subsequent call to gscBsiUtilEndTransaction() was unsuccessful, indicating that a transaction had not been started.
Status: Test 4.1 Failed."

Case 2: If the gscBsiUtilBeginTransaction() call returns the code BSI_NO_SPSSERVICE, then print
"gscBsiUtilBeginTransaction() is not supported.
Status: Test 4.1 Not Supported."

Case 3: If the gscBsiUtilBeginTransaction() call does not return the code BSI_OK or the code BSI_NO_SPSSERVICE, then print
"gscBsiUtilBeginTransaction() tested as a blocking transaction call, with no existing transaction lock, using valid parameters, returned an incorrect code.
Status: Test 4.1 Failed."

Test for Assertion 4.2

The function is tested as a non-blocking transaction call, with no existing transaction lock, using valid parameters.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiUtilBeginTransaction().

2. (Pre) Print "Testing of Assertion 4.2".

3. (Pre) Ensure that there is no existing transaction lock:

   Make a gscBsiUtilEndTransaction() call to the SPS, with
   - hCard == hCard0400.

   Case 1: If the gscBsiUtilEndTransaction() call returns either of the codes BSI_OK or BSI_NOT_TRANSACTED, then continue with 4.

   Case 2: If the gscBsiUtilEndTransaction() call does not return either of the codes BSI_OK or BSI_NOT_TRANSACTED, then print
   "It cannot be assured that there is no existing transaction lock. gscBsiUtilBeginTransaction() cannot be tested."
   End Test for Assertion 4.2.

4. Make a gscBsiUtilBeginTransaction() call to the SPS, with
   - hCard == hCard0400
   - blType == FALSE.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_OK or the return code BSI_NO_SPSSERVICE.

2. If the return code is BSI_OK, then a transaction is established with the smart card.

Perform this verification by issuing a call to gscBsiUtilEndTransaction().

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiUtilBeginTransaction() call returns the code BSI_OK, then verify that a transaction has indeed been started:

   Make a gscBsiUtilEndTransaction() call to the SPS, with
   - hCard == hCard0400.

   Case 1.1: If the gscBsiUtilEndTransaction() call returns the code BSI_OK, then print
   "gscBsiUtilBeginTransaction() tested as a non-blocking transaction call, with no existing transaction lock, using valid parameters, has been verified because a subsequent call to gscBsiUtilEndTransaction() was successful, indicating that a transaction had been started.
   Status: Test 4.2 Passed."

   Case 1.2: If the gscBsiUtilEndTransaction() call does not return the code BSI_OK, then print
   "gscBsiUtilBeginTransaction() tested as a non-blocking transaction call, with no existing transaction lock, using valid parameters, has not been verified because a subsequent call to gscBsiUtilEndTransaction() was unsuccessful, indicating that a transaction had not been started.
   Status: Test 4.2 Failed."

   Case 2: If the gscBsiUtilBeginTransaction() call returns the code BSI_NO_SPSSERVICE, then print
   "gscBsiUtilBeginTransaction() is not supported.
   Status: Test 4.2 Not Supported."

   Case 3: If the gscBsiUtilBeginTransaction() call does not return the code BSI_OK or the code BSI_NO_SPSSERVICE, then print
   "gscBsiUtilBeginTransaction() tested as a non-blocking transaction call, with no existing transaction lock, using valid parameters, returned an incorrect code.
   Status: Test 4.2 Failed."

Test for Assertion 4.3

The function is tested as a non-blocking transaction call, with another application having established a transaction lock, using valid parameters.

Until we encounter implementations that allow multiple simultaneous applications, I don't think we need to worry about this assertion.

Test for Assertion 4.4

The function is tested as a blocking transaction call, with another application having established a transaction lock, using valid parameters.

Until we encounter implementations that allow multiple simultaneous applications, I don't think we need to worry about this assertion.

Test for Assertion 4.5

The function is tested as a non-blocking transaction call using valid parameters, after the current application establishes a transaction lock.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiUtilBeginTransaction().

2. (Pre) Print "Testing of Assertion 4.5".

3. (Pre) Establish a transaction lock:

   Make a gscBsiUtilBeginTransaction() call to the SPS, with
   - hCard == hCard0400.
   - blType == FALSE.

   Case 1: If the gscBsiUtilBeginTransaction() call returns either of the codes BSI_OK or BSI_NOT_TRANSACTED, then continue with 4.

   Case 2: If the gscBsiUtilEndTransaction() call does not return either of the codes BSI_OK or BSI_NOT_TRANSACTED, then print
   "It cannot be assured that there is an existing transaction lock. gscBsiUtilBeginTransaction() cannot be tested."
   End Test for Assertion 4.5.

4. Make a gscBsiUtilBeginTransaction() call to the SPS, with
   - hCard == hCard0400
   - blType == FALSE.

Verification Goal:
To verify the Expected Results:

1. The call returns the return code BSI_NOT_TRANSACTED.

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiUtilBeginTransaction() call returns the code BSI_NOT_TRANSACTED, then:

   Perform Test for Assertion 9.4.5.1 using
   - hCard == hCard0400.

   Print
   "gscBsiUtilBeginTransaction(), tested as a non-blocking transaction call, using valid parameters, after the current application has established a transaction lock, has been verified.
   Status: Test 4.5 Passed."

   Case 2: If the gscBsiUtilBeginTransaction() call does not return the code BSI_NOT_TRANSACTED, then print
   "gscBsiUtilBeginTransaction(), tested as a non-blocking transaction call, using valid parameters, after the current application has established a transaction lock, returned an incorrect code.
   Status: Test 4.5 Failed."

Test for Assertion 4.6

The function is tested as a blocking transaction call using valid parameters, after the current application establishes a transaction lock.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiUtilBeginTransaction().

2. (Pre) Print "Testing of Assertion 4.6".

3. (Pre) Establish a transaction lock:

   Make a gscBsiUtilBeginTransaction() call to the SPS, with
   - hCard == hCard0400.
   - blType == FALSE.

   Case 1: If the gscBsiUtilBeginTransaction() call returns either of the codes BSI_OK or BSI_NOT_TRANSACTED, then continue with 4.

   Case 2: If the gscBsiUtilEndTransaction() call does not return either of the codes BSI_OK or BSI_NOT_TRANSACTED, then print
   "It cannot be assured that there is an existing transaction lock. gscBsiUtilBeginTransaction() cannot be tested."
   End Test for Assertion 4.6.

4. Make a gscBsiUtilBeginTransaction() call to the SPS, with
   - hCard == hCard0400
   - blType == TRUE.

Verification Goal:
To verify the Expected Results:

1. The call returns the return code BSI_NOT_TRANSACTED.

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiUtilBeginTransaction() call returns the code BSI_NOT_TRANSACTED, then:

   Perform Test for Assertion 9.4.6.1 using
   - hCard == hCard0400.

   Print
   "gscBsiUtilBeginTransaction(), tested as a blocking transaction call, using valid parameters, after the current application has established a transaction lock, has been verified.
   Status: Test 4.6 Passed."

   Case 2: If the gscBsiUtilBeginTransaction() call does not return the code BSI_NOT_TRANSACTED, then print
   "gscBsiUtilBeginTransaction(), tested as a blocking transaction call, using valid parameters, after the current application has established a transaction lock, returned an incorrect code.
   Status: Test 4.6 Failed."

Test for Assertion 4.7

The function is tested as a blocking transaction call, with no existing transaction lock, with a bad handle.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiUtilBeginTransaction().

2. (Pre) Print "Testing of Assertion 4.7".

3. (Pre) Ensure that there is no existing transaction lock:

   Make a gscBsiUtilEndTransaction() call to the SPS, with
   - hCard == hCard0400.

   Case 1: If the gscBsiUtilEndTransaction() call returns either of the codes BSI_OK or BSI_NOT_TRANSACTED, then continue with 4.

   Case 2: If the gscBsiUtilEndTransaction() call does not return either of the codes BSI_OK or BSI_NOT_TRANSACTED, then print
   "It cannot be assured that there is no existing transaction lock. gscBsiUtilBeginTransaction() cannot be tested."
   End Test for Assertion 4.7.

4. Make a gscBsiUtilBeginTransaction() call to the SPS, with
   - hCard /= hCard0400
   - blType == TRUE.

Verification Goal:

To verify the Expected Results:

1. The call returns

- the return code BSI_BAD_HANDLE or the return code BSI_NO_SPSSERVICE.

2. No transaction lock is established with the smart card.

Perform this verification by issuing a call to gscBsiUtilEndTransaction().

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiUtilBeginTransaction() call returns the code BSI_BAD_HANDLE, then:

   Perform Test for Assertion 9.4.7.1 using
   - hCard == hCard0400.

   Verify that no transaction lock has been established:

   Make a gscBsiUtilEndTransaction() call to the SPS, with
   - hCard == hCard0400.

   Case 1.1: If the gscBsiUtilEndTransaction() call returns the code BSI_NOT_TRANSACTED, then print
   "gscBsiUtilBeginTransaction(), tested as a blocking transaction call, with no existing transaction lock, with a bad handle, has been verified because a subsequent call to gscBsiUtilEndTransaction() was unsuccessful, indicating that a transaction had not been started.
   Status: Test 4.7 Passed."

   Case 1.2: If the gscBsiUtilEndTransaction() call returns the code BSI_OK, then print
   "gscBsiUtilBeginTransaction(), tested as a blocking transaction call, with no existing transaction lock, with a bad handle, has not been verified because a subsequent call to gscBsiUtilEndTransaction() was successful, indicating that a transaction had been started.
   Status: Test 4.7 Failed."

   Case 2: If the gscBsiUtilBeginTransaction() call returns the code BSI_NO_SPSSERVICE, then print
   "gscBsiUtilBeginTransaction() is not supported.
   Status: Test 4.7 Not Supported."

   Case 3: If the gscBsiUtilBeginTransaction() call does not return the code BSI_BAD_HANDLE or the code BSI_NO_SPSSERVICE, then print
   "gscBsiUtilBeginTransaction(), tested as a blocking transaction call, with no existing transaction lock, with a bad handle, returned an incorrect code.
   Status: Test 4.7 Failed."

5. gscBsiUtilEndTransaction()

Starting State for Each Test:

1. A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard0500.

Test for Assertion 5.1

The function is tested with an existing transaction lock, using valid parameters.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiUtilEndTransaction().

2. (Pre) Print "Testing of Assertion 5.1".

3. (Pre) Establish a transaction lock:

   Make a gscBsiUtilBeginTransaction() call to the SPS, with
   - hCard == hCard0500.
   - blType == TRUE.

   Case 1: If the gscBsiUtilBeginTransaction() call returns either of the codes BSI_OK or BSI_NOT_TRANSACTED, then continue with 4.

   Case 2: If the gscBsiUtilEndTransaction() call does not return either of the codes BSI_OK or BSI_NOT_TRANSACTED, then print "It cannot be assured that there is an existing transaction lock. gscBsiUtilBeginTransaction() cannot be tested."
   End Test for Assertion 5.1.

4. Make a gscBsiUtilEndTransaction() call to the SPS, with
   - hCard == hCard0500.

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_OK or the return code BSI_NO_SPSSERVICE.

2. If the return code is BSI_OK, then the previously existing transaction lock is ended.

Perform this verification by issuing a call to gscBsiUtilBeginTransaction().

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiUtilEndTransaction() call returns the code BSI_OK, then verify that the existing transaction has indeed ended:

   Make a gscBsiUtilBeginTransaction() call to the SPS, with
   - hCard == hCard0500
   - blType == TRUE.

Case 1.1: If the gscBsiUtilBeginTransaction() call returns the code BSI_OK, then print
"gscBsiUtilEndTransaction() called with an existing transaction lock and valid parameters has been verified because a subsequent call to gscBsiUtilBeginTransaction() was successful, indicating that the previous transaction had been ended.
Status: Test 5.1 Passed."

Case 1.2: If the gscBsiUtilEndTransaction() call does not return the code BSI_OK, then print
"gscBsiUtilEndTransaction() called with an existing transaction lock and valid parameters has not been verified because a subsequent call to gscBsiUtilBeginTransaction() was unsuccessful, indicating that the previous transaction had not been ended.
Status: Test 5.1 Failed."

Case 2: If the gscBsiUtilEndTransaction() call returns the code BSI_NO_SPSSERVICE, then print
"gscBsiUtilEndTransaction() is not supported.
Status: Test 5.1 Not Supported."

Case 3: If the gscBsiUtilBeginTransaction() call does not return the code BSI_OK or the code BSI_NO_SPSSERVICE, then print
"gscBsiUtilEndTransaction() called with an existing transaction lock and valid parameters returned an incorrect code.
Status: Test 5.1 Failed."

Test for Assertion 5.2

The function is tested with no existing transaction lock.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiUtilEndTransaction().

2. (Pre) Print "Testing of Assertion 5.2".

3. (Pre) Ensure that there is no existing transaction lock:

   Make a gscBsiUtilEndTransaction() call to the SPS, with
   - hCard == hCard0500.

   Case 1: If the gscBsiUtilEndTransaction() call returns either of the codes BSI_OK or BSI_NOT_TRANSACTED, then continue with 4.

   Case 2: If the gscBsiUtilEndTransaction() call does not return either of the codes BSI_OK or BSI_NOT_TRANSACTED, then print
   "It cannot be assured that there is no existing transaction lock. gscBsiUtilBeginTransaction() cannot be tested."
   End Test for Assertion 5.2.

4. Make a gscBsiUtilEndTransaction() call to the SPS, with

- hCard == hCard0500.

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_NOT_TRANSACTED or the return code BSI_NO_SPSSERVICE.

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiUtilEndTransaction() call returns the code BSI_NOT_TRANSACTED, then:

   Perform Test for Assertion 9.5.2.1 using
   - hCard == hCard0500.

   Print
   "gscBsiUtilEndTransaction() called with no existing transaction lock and valid parameters has been verified.
   Status: Test 5.2 Passed."

   Case 2: If the gscBsiUtilEndTransaction() call returns the code BSI_NO_SPSSERVICE, then print
   "gscBsiUtilEndTransaction() is not supported.
   Status: Test 5.2 Not Supported."

   Case 3: If the gscBsiUtilEndTransaction() call does not return the code BSI_NOT_TRANSACTED or the code BSI_NO_SPSSERVICE, then print
   "gscBsiUtilEndTransaction() called with no existing transaction lock and valid parameters returned an incorrect code.
   Status: Test 5.2 Failed."

Test for Assertion 5.3

The function is tested with an existing transaction lock, using a bad handle.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiUtilEndTransaction().

2. (Pre) Print "Testing of Assertion 5.3".

3. (Pre) Establish a transaction lock:

   Make a gscBsiUtilBeginTransaction() call to the SPS, with
   - hCard == hCard0500.
   - blType == TRUE.

   Case 1: If the gscBsiUtilBeginTransaction() call returns either of the codes BSI_OK or BSI_NOT_TRANSACTED, then continue with 4.

   Case 2: If the gscBsiUtilEndTransaction() call does not return either of the codes BSI_OK or BSI_NOT_TRANSACTED, then print

"It cannot be assured that there is an existing transaction lock. gscBsiUtilBeginTransaction() cannot be tested."

End Test for Assertion 5.3.

4. Make a gscBsiUtilEndTransaction() call to the SPS, with
   - hCard /= hCard0500.

Verification Goal:

To verify the Expected Results:

1. The call returns
   - the return code BSI_BAD_HANDLE or the return code BSI_NO_SPSSERVICE.

2. If the return code is BSI_BAD_HANDLE, then the previously existing transaction lock remains in effect.

Perform this verification by issuing a call to gscBsiUtilBeginTransaction().

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiUtilEndTransaction() call returns the code BSI_BAD_HANDLE, then:

   Perform Test for Assertion 9.5.3.1 using
   - hCard == hCard0500.

   Verify that the existing transaction is still in effect:

   Make a gscBsiUtilBeginTransaction() call to the SPS, with
   - hCard == hCard0500
   - blType == TRUE.

   Case 1.1: If the gscBsiUtilBeginTransaction() call returns the code BSI_NOT_TRANSACTED, then print
   "gscBsiUtilEndTransaction() called with an existing transaction lock and a bad handle has been verified because a subsequent call to gscBsiUtilBeginTransaction() was unsuccessful, indicating that the previous transaction had not been ended.
   Status: Test 5.3 Passed."

   Case 1.2: If the gscBsiUtilEndTransaction() call returns the code BSI_OK, then print
   "gscBsiUtilEndTransaction() called with an existing transaction lock and a bad handle has not been verified because a subsequent call to gscBsiUtilBeginTransaction() was successful, indicating that the previous transaction had been ended.
   Status: Test 5.3 Failed."

   Case 2: If the gscBsiUtilEndTransaction() call returns the code BSI_NO_SPSSERVICE, then print
   "gscBsiUtilEndTransaction() is not supported.
   Status: Test 5.3 Not Supported."

   Case 3: If the gscBsiUtilBeginTransaction() call does not return the code BSI_BAD_HANDLE or the code BSI_NO_SPSSERVICE, then print

"gscBsiUtilEndTransaction() called with an existing transaction lock and a bad handle returned an incorrect code.
Status: Test 5.3 Failed."

6. gscBsiUtilGetVersion()

Test for Assertion 6.1

The function is tested using valid parameters (Discovery Method 1).

Instantiation Scenario:

1. (Pre) Allocate the unsigned long* punVersionLen0601 == 0.

2. (Pre) Print "Testing of Assertion 6.1".

3. Make a gscBsiUtilGetVersion() call to the SPS using
   - uszVersion == NULL
   - punVersionLen == punVersionLen0601.

   Case 1: If the gscBsiUtilGetVersion() call returns the code BSI_OK, then:

   Print
   "gscBsiUtilGetVersion() (Discovery Method 1, Discovery Mode) succeeded."

   Continue with 4.

   Case 2: If the gscBsiUtilGetVersion() call does not return the code BSI_OK, then print
   "gscBsiUtilGetVersion() (Discovery Method 1, Discovery Mode) did not return the code BSI_OK.
   Status: Test 6.1 Failed.
   End Test for Assertion 6.1.

4. Allocate the unsigned char* uszVersion0601 with length == punVersionLen0601*.

5. Make a gscBsiUtilGetVersion() call to the SPS using
   - uszVersion == uszVersion0601
   - punVersionLen == punVersionLen0601.

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_OK
   - uszVersion0601 == the BSI implementation version of the SPS.

Perform this verification by inspection.

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiUtilGetVersion() call returns the code BSI_OK, then manually inspect uszVersion0601.
   Case 1.1: If uszVersion0601 represents the BSI implementation version of the SPS, then print
   "gscBsiUtilGetVersion() (Discovery Method 1, Final Mode) called with valid parameters has been verified by inspection.
   Status: Test 6.1 Passed."

Case 1.2: If uszVersion0601 does not represent the BSI implementation version of the SPS, then print
"gscBsiUtilGetVersion() (Discovery Method 1, Final Mode) called with valid parameters has not been verified by inspection.
Status: Test 6.1 Failed."

Case 2: If the gscBsiUtilGetVersion() call does not return the code BSI_OK, then print
"gscBsiUtilGetVersion() (Discovery Method 1, Final Mode) called with valid parameters returned an incorrect code.
Status: Test 6.1 Failed.

Test for Assertion 6.2

The function is tested using valid parameters (Discovery Method 2).

Instantiation Scenario:

1. (Pre) Allocate the unsigned char* uszVersion0602 with length 0.

2. (Pre) Allocate the unsigned long* punVersionLen0602 == 0.

3. (Pre) Print "Testing of Assertion 6.2".

4. Make a gscBsiUtilGetVersion() call to the SPS using
   - uszVersion == uszVersion0602
   - punVersionLen == punVersionLen0602.

   Case 1: If the gscBsiUtilGetVersion() call returns the code BSI_INSUFFICIENT_BUFFER, then:

   Perform Test for Assertion 9.6.2.1.

   Print
   "gscBsiUtilGetVersion() (Discovery Method 2, Discovery Mode) correctly returned the code BSI_INSUFFICIENT_BUFFER."

   Continue with 5.

   Case 2: If the gscBsiUtilGetVersion() call does not return the code BSI_INSUFFICIENT_BUFFER, then print
   "gscBsiUtilGetVersion() (Discovery Method 2, Discovery Mode) did not return the code BSI_INSUFFICIENT_BUFFER.
   Status: Test 6.2 Failed.
   End Test for Assertion 6.2.

5. Re-allocate the unsigned char* uszVersion0602 to be of length == punVersionLen0602*.

6. Make a gscBsiUtilGetVersion() call to the SPS using
   - uszVersion == uszVersion0602
   - punVersionLen == punVersionLen0602.

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_OK
   - uszVersion0602 == the BSI implementation version of the SPS.

Perform this verification by inspection.

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiUtilGetVersion() call returns the code BSI_OK, then manually inspect uszVersion0602.

   Case 1.1: If uszVersion0602 represents the BSI implementation version of the SPS, then print
   "gscBsiUtilGetVersion() (Discovery Method 2, Final Mode) called with valid parameters has been verified by inspection.
   Status: Test 6.2 Passed."

   Case 1.2: If uszVersion0602 does not represent the BSI implementation version of the SPS, then print
   "gscBsiUtilGetVersion() (Discovery Method 2, Final Mode) called with valid parameters has not been verified by inspection.
   Status: Test 6.2 Failed."

   Case 2: If the gscBsiUtilGetVersion() call does not return the code BSI_OK, then print
   "gscBsiUtilGetVersion() (Discovery Method 2, Final Mode) called with valid parameters returned an incorrect code.
   Status: Test 6.2 Failed."

Test for Assertion 6.3

The function is tested using a bad Version length.

Instantiation Scenario:

1. (Pre) Allocate the unsigned long* punVersionLen0603 == 0.

2. (Pre) Print "Testing of Assertion 6.3".

3. Make a gscBsiUtilGetVersion() call to the SPS using
   - uszVersion == NULL
   - punVersionLen == punVersionLen0603.

   Case 1: If the gscBsiUtilGetVersion() call returns the code BSI_OK, then:

   Print
   "gscBsiUtilGetVersion() (Discovery Method 1, Discovery Mode) succeeded."

   Continue with 4.

   Case 2: If the gscBsiUtilGetVersion() call does not return the code BSI_OK, then:

   Print
   "gscBsiUtilGetVersion() (Discovery Method 1, Discovery Mode) did not return the code BSI_OK.

End Test for Assertion 6.3.

4. Allocate the unsigned char* uszVersion0603 with length == punVersionLen0603*.

5. Assign punVersionLen0603 == 0.

6. Make a gscBsiUtilGetVersion() call to the SPS using
   - uszVersion == uszVersion0603
   - punVersionLen == punVersionLen0603.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_BAD_PARAM.

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiUtilGetVersion() call returns the code BSI_BAD_PARAM, then:

   Print
   "gscBsiUtilGetVersion() (Discovery Method 1, Final Mode) called with a bad Version length has been verified.
   Status: Test 6.3 Passed."

   Case 2: If the gscBsiUtilGetVersion() call does not return the code BSI_BAD_PARAM, then:

   Print

   "gscBsiUtilGetVersion() (Discovery Method 1, Final Mode) called with a bad Version length returned an incorrect code.
   Status: Test 6.3 Failed.

7. gscBsiUtilGetCardProperties()

Starting State for each Test:

1. A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard0700.

2. There exists an unsigned long* punCardCapability0700.

Test for Assertion 7.1

The function is tested using valid parameters (Discovery Method 1).

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiUtilGetCardProperties().

2. (Pre) Allocate the unsigned long* punCCCUniqueIDLen0701 == 0.

3. (Pre) Print "Testing of Assertion 7.1".

4. Make a gscBsiUtilGetCardProperties() call to the SPS, using
   - hCard == hCard0700
   - uszCCCUniqueID == NULL
   - punCCCUniqueIDLen == punCCCUniqueIDLen0701
   - punCardCapability == punCardCapability0700.

   Case 1: If the gscBsiUtilGetCardProperties() call returns the code BSI_OK, then print
   "gscBsiUtilGetCardProperties() (Discovery Method 1, Discovery Mode) succeeded."
   Continue with 5.

   Case 2: If the gscBsiUtilGetCardProperties() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiUtilGetCardProperties() is not supported.
   Status: Test 7.1 Not Supported."
   End Test for Assertion 7.1.

   Case 3: If the gscBsiUtilGetCardProperties() call does not return the code BSI_OK or the code BSI_NO_CARDSERVICE, then print
   "gscBsiUtilGetCardProperties() (Discovery Method 1, Discovery Mode) returned an incorrect code.
   Status: Test 7.1 Failed."
   End Test for Assertion 7.1.

5. Allocate the unsigned char* uszCCCUniqueID0701 with length == punCCCUniqueIDLen0701.

6. Make a gscBsiUtilGetCardProperties() call to the SPS, using
   - hCard == hCard0700
   - uszCCCUniqueID == uszCCCUniqueID0701
   - punCCCUniqueIDLen == punCCCUniqueIDLen0701
   - punCardCapability == punCardCapability0700.

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_OK
   - uszCCCUniqueID0701 == the Card Capability Container ID
   - punCardCapability0700 == one of the recognized bitwise masks identifying the provider of the connected card.

Perform this verification by inspection.

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiUtilGetCardProperties() call returns the code BSI_OK, then manually inspect uszCCCUniqueID0701 and punCardCapability0700.

   Case 1.1: If CCCUniqueID0701 is the Card Capability Container ID and punCardCapability0700 is one of
   - 00000001 (hex)
   - 00000002 (hex)
   - 00000004 (hex)

   then print
   "gscBsiUtilGetCardProperties() (Discovery Method 1, Final Mode) called with valid parameters has been verified by inspection.
   Status: Test 7.1 Passed."

   Case 1.2: If either uszCCCUniqueID0701 is not the Card Capability Container ID or punCardCapability0700 is not one of the above masks, then print
   "gscBsiUtilGetCardProperties() (Discovery Method 1, Final Mode) called with valid parameters has not been verified by inspection.
   Status: Test 7.1 Failed."

   Case 2: If the gscBsiUtilGetCardProperties() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiUtilGetCardProperties() is not supported.
   Status: Test 7.1 Not Supported."

   Case 3: If the gscBsiUtilGetCardProperties() call does not return the code BSI_OK, then print
   "gscBsiUtilGetCardProperties() (Discovery Method 1, Final Mode) called with valid parameters returned an incorrect code.
   Status: Test 7.1 Failed."

Test for Assertion 7.2

The function is tested using valid parameters (Discovery Method 2).

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiUtilGetCardProperties().

2. (Pre) Allocate the unsigned char* uszCCCUniqueID0702 with length 0.

3. (Pre) Allocate the unsigned long* punCCCUniqueIDLen0702 == 0.

4. (Pre) Print "Testing of Assertion 7.2".

5. Make a gscBsiUtilGetCardProperties() call to the SPS, using
   - hCard == hCard0700
   - uszCCCUniqueID == uszCCCUniqueID0702
   - punCCCUniqueIDLen == punCCCUniqueIDLen0702
   - punCardCapability == punCardCapability0700.

   Case 1: If the gscBsiUtilGetCardProperties() call returns the code BSI_INSUFFICIENT_BUFFER, then print
   "gscBsiUtilGetCardProperties() (Discovery Method 2, Discovery Mode) correctly returned the code BSI_INSUFFICIENT_BUFFER."
   Continue with 6.

   Case 2: If the gscBsiUtilGetCardProperties() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiUtilGetCardProperties() is not supported.
   Status: Test 7.2 Not Supported."
   End Test for Assertion 7.2.

   Case 3: If the gscBsiUtilGetCardProperties() call does not return the code BSI_INSUFFICIENT_BUFFER or the code BSI_NO_CARDSERVICE, then print
   "gscBsiUtilGetCardProperties() (Discovery Method 2, Discovery Mode) returned an incorrect code.
   Status: Test 7.2 Failed.
   End Test for Assertion 7.2.

6. Allocate the unsigned char* uszCCCUniqueID0702 with length == punCCCUniqueIDLen0702.

7. Make a gscBsiUtilGetCardProperties() call to the SPS, using
   - hCard == hCard0700
   - uszCCCUniqueID == uszCCCUniqueID0702
   - punCCCUniqueIDLen == punCCCUniqueIDLen0702
   - punCardCapability == punCardCapability0700.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_OK
   - uszCCCUniqueID0702 == the Card Capability Container ID
   - punCardCapability0700 == one of the recognized bitwise masks identifying the provider of the connected card.

Perform this verification by inspection.

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiUtilGetCardProperties() call returns the code BSI_OK, then manually inspect uszCCCUniqueID0702 and punCardCapability0700.

   Case 1.1: If CCCUniqueID0702 is the Card Capability Container ID and cardCapability0700 is one of
   - 00000001 (hex)

- 00000002 (hex)
- 00000004 (hex)

then print
"gscBsiUtilGetCardProperties() (Discovery Method 2, Final Mode) called with valid parameters has been verified by inspection.
Status: Test 7.2 Passed."

Case 1.2: If either uszCCCUniqueID0701 is not the Card Capability Container ID or cardCapability0700 is not one of the above masks, then print
"gscBsiUtilGetCardProperties() (Discovery Method 2, Final Mode) called with valid parameters has not been verified by inspection.
Status: Test 7.2 Failed."

Case 2: If the gscBsiUtilGetCardProperties() call returns the code BSI_NO_CARDSERVICE, then print
"gscBsiUtilGetCardProperties() is not supported.
Status: Test 7.2 Not Supported."

Case 3: If the gscBsiUtilGetCardProperties() call does not return the code BSI_OK, then print
"gscBsiUtilGetCardProperties() (Discovery Method 2, Final Mode) called with valid parameters returned an incorrect code.
Status: Test 7.2 Failed."

Test for Assertion 7.3

The function is tested using a bad handle (Discovery Method 1, Discovery Mode).

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiUtilGetCardProperties().

2. (Pre) Allocate the unsigned long* punCCCUniqueIDLen0703 == 0.

3. (Pre) Print "Testing of Assertion 7.3".

4. Make a gscBsiUtilGetCardProperties() call to the SPS, using
   - hCard /= hCard0700
   - uszCCCUniqueID == NULL
   - punCCCUniqueIDLen == punCCCUniqueIDLen0703
   - punCardCapability == punCardCapability0700.

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_BAD_HANDLE.

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiUtilGetCardProperties() call returns the code BSI_BAD_HANDLE, then:

   Perform Test for Assertion 9.7.3.1 using

- hCard == hCard0700.

Print
"gscBsiUtilGetCardProperties() (Discovery Method 1, Discovery Mode) called with a bad handle has been verified.
Status: Test 7.3 Passed."

Case 2: If the gscBsiUtilGetCardProperties() call returns the code BSI_NO_CARDSERVICE, then print
"gscBsiUtilGetCardProperties() is not supported.
Status: Test 7.3 Not Supported."

Case 3: If the gscBsiUtilGetCardProperties() call does not return the code BSI_BAD_HANDLE or the code BSI_NO_CARDSERVICE, then print
"gscBsiUtilGetCardProperties() (Discovery Method 1, Discovery Mode) called with a bad handle returned an incorrect code.
Status: Test 7.3 Failed."

Test for Assertion 7.4

The function is tested using a bad handle (Discovery Method 1, Final Mode).

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiUtilGetCardProperties().

2. (Pre) Allocate the unsigned long* punCCCUniqueIDLen0704 == 0.

3. (Pre) Make a gscBsiUtilGetCardProperties() call to the SPS, using
   - hCard == hCard0700
   - uszCCCUniqueID == NULL
   - punCCCUniqueIDLen == punCCCUniqueIDLen0704
   - punCardCapability == punCardCapability0700.

4. (Pre) Allocate the unsigned char* uszCCCUniqueID0704 with length == punCCCUniqueIDLen0704.

5. (Pre) Print "Testing of Assertion 7.4".

6. Make a gscBsiUtilGetCardProperties() call to the SPS, using
   - hCard /= hCard0700
   - uszCCCUniqueID == uszCCCUniqueID0704
   - punCCCUniqueIDLen == punCCCUniqueIDLen0704
   - punCardCapability == punCardCapability0700.

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_BAD_HANDLE.

Verification and Reporting Scenario:

1. Case **1**: If the gscBsiUtilGetCardProperties() call returns the code BSI_BAD_HANDLE, then:

Perform Test for Assertion 9.7.3.1 using
- hCard == hCard0700.

Print
"gscBsiUtilGetCardProperties() (Discovery Method 1, Final Mode) called with a bad handle has been verified.
Status: Test 7.4 Passed."

Case 2: If the gscBsiUtilGetCardProperties() call returns the code BSI_NO_CARDSERVICE, then print
"gscBsiUtilGetCardProperties() is not supported.
Status: Test 7.4 Not Supported."

Case 3: If the gscBsiUtilGetCardProperties() call does not return the code BSI_BAD_HANDLE or the code BSI_NO_CARDSERVICE, then print
"gscBsiUtilGetCardProperties() (Discovery Method 1, Final Mode) called with a bad handle returned an incorrect code.
Status: Test 7.4 Failed."

Test for Assertion 7.5

The function is tested with a removed card (Discovery Method 1, Discovery Mode).

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiUtilGetCardProperties().

2. (Pre) Allocate the unsigned long* punCCCUniqueIDLen0705 == 0.

3. (Pre) Print "Testing of Assertion 7.5".

4. Remove the connected card from the reader.

5. Make a gscBsiUtilGetCardProperties() call to the SPS, using
   - hCard == hCard0700
   - uszCCCUniqueID == NULL
   - punCCCUniqueIDLen == punCCCUniqueIDLen0705
   - punCardCapability == punCardCapability0700.

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_CARD_REMOVED.

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiUtilGetCardProperties() call returns the code BSI_CARD_REMOVED, then:

   Print
   "gscBsiUtilGetCardProperties() (Discovery Method 1, Discovery Mode) called with the connected card removed has been verified.
   Status: Test 7.5 Passed."

Case 2: If the gscBsiUtilGetCardProperties() call returns the code BSI_NO_CARDSERVICE, then print
"gscBsiUtilGetCardProperties() is not supported.
Status: Test 7.5 Not Supported."

Case 3: If the gscBsiUtilGetCardProperties() call does not return the code BSI_CARD_REMOVED or the code BSI_NO_CARDSERVICE, then print
"gscBsiUtilGetCardProperties() (Discovery Method 1, Discovery Mode) called with the connected card removed returned an incorrect code.
Status: Test 7.5 Failed."

Test for Assertion 7.6

The function is tested with a removed card (Discovery Method 1, Final Mode).

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiUtilGetCardProperties().

2. (Pre) Allocate the unsigned long* punCCCUniqueIDLen0706 == 0.

3. (Pre) Make a gscBsiUtilGetCardProperties() call to the SPS, using
   - hCard == hCard0700
   - uszCCCUniqueID == NULL
   - punCCCUniqueIDLen == punCCCUniqueIDLen0706
   - punCardCapability == punCardCapability0700.

4. (Pre) Allocate the unsigned char* uszCCCUniqueID0706 with length == punCCCUniqueIDLen0706.

5. (Pre) Print "Testing of Assertion 7.6".

6. Remove the connected card from the reader.

7. Make a gscBsiUtilGetCardProperties() call to the SPS, using
   - hCard == hCard0700
   - uszCCCUniqueID == uszCCCUniqueID0706
   - punCCCUniqueIDLen == punCCCUniqueIDLen0706
   - punCardCapability == punCardCapability0700.

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_CARD_REMOVED.

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiUtilGetCardProperties() call returns the code BSI_CARD_REMOVED, then print
   "gscBsiUtilGetCardProperties() (Discovery Method 1, Final Mode) called with the connected card removed has been verified.
   Status: Test 7.6 Passed."

Case 2: If the gscBsiUtilGetCardProperties() call returns the code BSI_NO_CARDSERVICE, then print
"gscBsiUtilGetCardProperties() is not supported.
Status: Test 7.6 Not Supported."

Case 3: If the gscBsiUtilGetCardProperties() call does not return the code BSI_CARD_REMOVED or the code BSI_NO_CARDSERVICE, then print
"gscBsiUtilGetCardProperties() (Discovery Method 1, Final Mode) called with the connected card removed returned an incorrect code.
Status: Test 7.6 Failed."

Test for Assertion 7.7

The function is tested with another application having established a transaction lock.

Until we encounter implementations that allow multiple simultaneous applications, I don't think we need to worry about this assertion.

Test for Assertion 7.8

The function is tested using a bad unique ID length (Discovery Method 1, Final Mode).

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiUtilGetCardProperties().

2. (Pre) Allocate the unsigned long* punCCCUniqueIDLen0708 == 0.

3. (Pre) Make a gscBsiUtilGetCardProperties() call to the SPS, using
   - hCard == hCard0700
   - uszCCCUniqueID == NULL
   - punCCCUniqueIDLen == punCCCUniqueIDLen0708
   - punCardCapability == punCardCapability0700.

4. (Pre) Allocate the unsigned char* uszCCCUniqueID0704 with length == punCCCUniqueIDLen0708.

5. (Pre) Assign punCCCUniqueIDLen0708 == 0.

6. (Pre) Print "Testing of Assertion 7.8".

7. Make a gscBsiUtilGetCardProperties() call to the SPS, using
   - hCard == hCard0700
   - uszCCCUniqueID == uszCCCUniqueID0708
   - punCCCUniqueIDLen == punCCCUniqueIDLen0708
   - punCardCapability == punCardCapability0700.

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_BAD_PARAM.

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiUtilGetCardProperties() call returns the code BSI_BAD_PARAM, then:

   Perform Test for Assertion 9.7.8.1 using
   - hCard == hCard0700.

   Print
   "gscBsiUtilGetCardProperties() (Discovery Method 1, Final Mode) called with a bad unique ID length has been verified.
   Status: Test 7.8 Passed."

   Case 2: If the gscBsiUtilGetCardProperties() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiUtilGetCardProperties() is not supported.
   Status: Test 7.8 Not Supported."

   Case 3: If the gscBsiUtilGetCardProperties() call does not return the code BSI_BAD_PARAM or the code BSI_NO_CARDSERVICE, then print
   "gscBsiUtilGetCardProperties() (Discovery Method 1, Final Mode) called with a bad unique ID length returned an incorrect code.
   Status: Test 7.8 Failed."

8. gscBsiUtilGetCardStatus()

Starting State for Each Test:

1. A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard0800.

Test for Assertion 8.1

The function is tested using valid parameters.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiUtilGetCardStatus().

2. (Pre) Print "Testing of Assertion 8.1".

3. Make a gscBsiUtilGetCardStatus() call to the SPS, using
   - hCard == hCard0800.

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_OK.

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiUtilGetCardStatus() call returns the code BSI_OK, then print
   "gscBsiUtilGetCardStatus() called with valid parameters has been verified.
   Status: Test 8.1 Passed."

   Case 2: If the gscBsiUtilGetCardStatus() call does not return the code BSI_OK, then print
   "gscBsiUtilGetCardStatus() called with valid parameters returned an incorrect code.
   Status: Test 8.1 Failed."

Test for Assertion 8.2

The function is tested using a bad handle.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiUtilGetCardStatus().

2. (Pre) Print "Testing of Assertion 8.2".

3. Make a gscBsiUtilGetCardStatus() call to the SPS, using
   - hCard /= hCard0800.

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_BAD_HANDLE.

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiUtilGetCardStatus() call returns the code BSI_BAD_HANDLE, then:

   Perform Test for Assertion 9.8.2.1 using
   - hCard == hCard0900.

   Print
   "gscBsiUtilGetCardStatus() called with a bad handle has been verified.
   Status: Test 8.2 Passed."

   Case 2: If the gscBsiUtilGetCardStatus() call does not return the code BSI_BAD_HANDLE, then print
   "gscBsiUtilGetCardStatus() called with a bad handle returned an incorrect code.
   Status: Test 8.2 Failed."

Test for Assertion 8.3

The function is tested with a card that has been removed.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiUtilGetCardStatus().

2. (Pre) Print "Testing of Assertion 8.3".

3. Remove the connected card from the reader.

4. Make a gscBsiGetCardStatus() call to the SPS, with
   - hCard == hCard0800.

Verification Goal:

To verify the Expected Results:

1. The call returns
   - the return code BSI_CARD_REMOVED.

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiUtilGetCardStatus() call returns the code BSI_CARD_REMOVED, then print
   "gscBsiUtilGetCardStatus() called with the connected card removed has been verified.
   Status: Test 8.3 Passed."

   Case 2: If the gscBsiUtilGetCardStatus() call does not return the code BSI_CARD_REMOVED, then print
   "gscBsiUtilGetCardStatus() called with the connected card removed returned an incorrect code.
   Status: Test 8.3 Failed.

9. gscBsiUtilGetExtendedErrorText()

Test for Assertion 9.X.Y.1

The function is tested using valid parameters.

Instantiation Scenario:

1. (Pre) Allocate the char* uszErrorTextXY with length 256.

2. (Pre) Print "Testing of Assertion 9.X.Y.1".

3. Make a gscBsiUtilGetExtendedErrorText call to the SPS, using
   - hCard == hCardXY
   - uszErrorText == uszErrorTextXY.

Verification Goal:
To verify the Expected Results:

1. The call returns
   - either the return code BSI_OK or the return code BSI_NO_TEXT_AVAILABLE
   - if BSI_OK is the code returned, then uszErrorTextXY == an extended error message
   - if BSI_NO_TEXT_AVAILABLE is the code returned, then uszErrorTextXY== "".

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiGetExtendedErrorText() call returns the code BSI_OK, then print
   "gscBsiUtilGetExtendedErrorText() called with valid parameters has been verified (text returned).
   Status: Test 9.X.Y.1 Passed."

   Case 2: If the gscBsiGetExtendedErrorText() call returns the code BSI_NO_TEXT_AVAILABLE, then print
   "gscBsiUtilGetExtendedErrorText() called with valid parameters has been verified (no text available).
   Status: Test 9.X.Y.1 Passed."

   Case 3: If the gscBsiGetExtendedErrorText() call returns a code other than BSI_OK or BSI_NO_TEXT_AVAILABLE, then print
   "gscBsiUtilGetExtendedErrorText() called with valid parameters returned an incorrect code.
   Status: Test 9.X.Y.1 Failed."

10. gscBsiUtilGetReaderList()

Test for Assertion 10.1

The function is tested using valid parameters (Discovery Method 1).

Instantiation Scenario:

1. (Pre) Allocate the unsigned long* punReaderListLen1001 == 0.

2. (Pre) Print "Testing of Assertion 10.1".

3. Make a gscBsiUtilGetReaderList() call to the SPS, with
   - uszReaderList == NULL
   - punReaderListLen == punReaderListLen1001.

   Case 1: If the gscBsiUtilGetReaderList() call returns the code BSI_OK, then print
   "gscBsiUtilGetReaderList() (Discovery Method 1, Discovery Mode) succeeded."
   Continue with 4.

   Case 2: If the gscBsiUtilGetReaderList() call does not return the code BSI_OK, then print
   "gscBsiUtilGetReaderList() (Discovery Method 1, Discovery Mode) did not return the code BSI_OK.
   Status: Test 10.1 Failed.
   End Test for Assertion 10.1.

4. Allocate the unsigned char* uszReaderList1001 with length == punReaderListLen1001.

5. Make a gscBsiUtilGetReaderList() call to the SPS, using
   - uszReaderList == uszReaderList1001
   - punReaderListLen == punReaderListLen1001.

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_OK
   - uszReaderList1001 == a list of available readers.

Perform this verification by inspection.

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiUtilGetReaderList() call returns the code BSI_OK, then manually inspect uszReaderList1001.

   Case 1.1: If uszReaderList1001 is a list of available printers, then print
   "gscBsiUtilGetReaderList() (Discovery Method 1, Final Mode) called with valid parameters has been verified by inspection.
   Status: Test 10.1 Passed."

   Case 1.2: If uszReaderList1001 is not a list of available readers, then print

"gscBsiUtilGetReaderList() (Discovery Method 1, Final Mode) called with valid parameters has not been verified by inspection.
Status: Test 10.1 Failed."

Case 2: If the gscBsiUtilGetReaderList() call does not return the code BSI_OK, then print
"gscBsiUtilGetReaderList() (Discovery Method 1, Final Mode) called with valid parameters returned an incorrect code.
Status: Test 10.1 Failed."

Test for Assertion 10.2

The function is tested using valid parameters (Discovery Method 2).

Instantiation Scenario:

1. (Pre) Allocate the unsigned char* uszReaderList1002 with length 0.

2. (Pre) Allocate the unsigned long* punReaderListLen1002 == 0.

3. (Pre) Print "Testing of Assertion 10.2".

4. Make a gscBsiUtilGetReaderList() call to the SPS, with
   - uszReaderList == uszReaderList1002
   - punReaderListLen == punReaderListLen1002.

   Case 1: If the gscBsiUtilGetReaderList() call returns the code BSI_INSUFFICIENT_BUFFER, then print
   "gscBsiUtilGetReaderList() (Discovery Method 2, Discovery Mode) correctly returned the code BSI_INSUFFICIENT_BUFFER."
   Continue with 5.

   Case 2: If the gscBsiUtilGetReaderList() call does not return the code BSI_INSUFFICIENR_BUFFER, then print
   "gscBsiUtilGetReaderList() (Discovery Method 2, Discovery Mode) did not return the code BSI_INSUFFICIENT_BUFFER.
   Status: Test 10.2 Failed.
   End Test for Assertion 10.2.

5. Re-allocate the unsigned char* uszReaderList1002 with length == punReaderListLen1002.

6. Make a gscBsiUtilGetReaderList() call to the SPS, using
   - uszReaderList == uszReaderList1002
   - punReaderListLen == punReaderListLen1002.

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_OK
   - uszReaderList1002 == a list of available readers.

Perform this verification by inspection.

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiUtilGetReaderList() call returns the code BSI_OK, then manually inspect uszReaderList1002.

   Case 1.1: If uszReaderList1002 is a list of available printers, then print
   "gscBsiUtilGetReaderList() (Discovery Method 2, Final Mode) called with valid parameters has been verified by inspection.
   Status: Test 10.2 Passed."

   Case 1.2: If uszReaderList1002 is not a list of available readers, then print
   "gscBsiUtilGetReaderList() (Discovery Method 2, Final Mode) called with valid parameters has not been verified by inspection.
   Status: Test 10.2 Failed."

   Case 2: If the gscBsiUtilGetReaderList() call does not return the code BSI_OK, then print
   "I"gscBsiUtilGetReaderList() (Discovery Method 2, Final Mode) called with valid parameters returned an incorrect code.
   Status: Test 10.2 Failed."

Test for Assertion 10.3

The function is tested using a bad ReaderList length (Discovery Method 1, Final Mode).

Instantiation Scenario:

1. (Pre) Allocate the unsigned long* punReaderListLen1003 == 0.

2. (Pre) Print "Testing of Assertion 10.3".

3. Make a gscBsiUtilGetReaderList() call to the SPS, with
   - uszReaderList == NULL
   - punReaderListLen == punReaderListLen1003.

   Case 1: If the gscBsiUtilGetReaderList() call returns the code BSI_OK, then print
   "gscBsiUtilGetReaderList() (Discovery Method 1, Discovery Mode) succeeded."
   Continue with 4.

   Case 2: If the gscBsiUtilGetReaderList() call does not return the code BSI_OK, then print
   "gscBsiUtilGetReaderList() (Discovery Method 1, Discovery Mode) did not return the code BSI_OK.
   Status: Test 10.3 Failed.
   End Test for Assertion 10.3.

4. Allocate the unsigned char* uszReaderList1003 with length == punReaderListLen1003.

5. Assign punReaderListLen1003 == 0.

6. Make a gscBsiUtilGetReaderList() call to the SPS, using
   - uszReaderList == uszReaderList1003

- punReaderListLen == punReaderListLen1003.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_BAD_PARAM.

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiUtilGetReaderList() call returns the code BSI_BAD_PARAM, then print
   "gscBsiUtilGetReaderList() (Discovery Method 1, Final Mode) called with a bad ReaderList length has been verified. **Status**: Test 10.3 Passed."

   Case 2: If the gscBsiUtilGetReaderList() call does not return the code BSI_BAD_PARAM, then print
   "gscBsiUtilGetReaderList() (Discovery Method 1, Final Mode) called with a bad ReaderList length returned an incorrect code. **Status**: Test 10.3 Failed."

11. gscBsiUtilPassthru()

Starting State for each Test:

1. A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1100.

Test for Assertion 11.1

The function is tested using valid parameters (Discovery Method 1).

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiUtilPassthru().

2. (Pre) Allocate the unsigned long* punCardResponseLen1101 == 0.

3. (Pre) Print "Testing of Assertion 11.1".

4. Make a gscBsiUtilPassthru() call to the SPS, using
   - hCard == hCard1100
   - uszCardCommand == _goodCardCommand
   - unCardCommandLen == _goodCardCommandLen
   - uszCardResponse == NULL
   - punCardResponseLen == punCardResponseLen1101.

   Case 1: If the gscBsiUtilPassthru() call returns the code BSI_OK, then print
   "gscBsiUtilPassthru() (Discovery Method 1, Discovery Mode) succeeded."
   Continue with 5.

   Case 2: If the gscBsiUtilPassthru() call does not return the code BSI_OK, then print
   "gscBsiUtilPassthru() (Discovery Method 1, Discovery Mode) did not return the code BSI_OK.
   Status: Test 11.1 Failed."
   End Test for Assertion 11.1.

5. Allocate the unsigned char* uszCardResponse1101 with length == punCardResponseLen1101.

6. Make a gscBsiUtilPassthru() call to the SPS, using
   - hCard == hCard1100
   - uszCardCommand == _goodCardCommand
   - unCardCommandLen == _goodCardCommandLen
   - uszCardResponse == uszCardResponse1101
   - punCardResponseLen == punCardResponseLen1101.

Verification Goal:

To verify the Expected Results:

1. The call returns
   - the return code BSI_OK
   - uszCardResponse1101 == a string containing the APDU response from the connected card.

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiUtilPassthru() call returns the code BSI_OK, then:

   Case 1.1: If uszCardResponse1101 == one of the elements of _goodCardResponse, then print
   "gscBsiUtilPassthru() (Discovery Method 1, Final Mode) called with valid parameters has been verified.
   Status: Test 11.1 Passed."

   Case 1.2: If uszCardResponse1101 /= one of the elements of _goodCardResponse, then print
   "gscBsiUtilPassthru() (Discovery Method 1, Final Mode) called with valid parameters has not been verified.
   Status: Test 11.1 Failed."

   Case 2: If the gscBsiUtilPassthru() call does not return the code BSI_OK, then print
   "gscBsiUtilPassthru() (Discovery Method 1, Final Mode) called with valid parameters returned an incorrect code.
   Status: Test 11.1 Failed."

Test for Assertion 11.2

The function is tested using valid parameters (Discovery Method 2).

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiUtilPassthru().

2. (Pre) Allocate the unsigned char* uszCardResponse1102 with length 0.

3. (Pre) Allocate the unsigned long* punCardResponseLen1102 == 0.

4. (Pre) Print "Testing of Assertion 11.2".

5. Make a gscBsiUtilPassthru() call to the SPS, using
   - hCard == hCard1100
   - uszCardCommand == _goodCardCommand
   - unCardCommandLen == _goodCardCommandLen
   - uszCardResponse == uszCardResponse1102
   - punCardResponseLen == punCardResponseLen1102.

   Case 1: If the gscBsiUtilPassthru() call returns the code BSI_INSUFFICIENT_BUFFER, then print
   "gscBsiUtilPassthru() (Discovery Method 2, Discovery Mode) correctly returned the code BSI_INSUFFICIENT_BUFFER."
   Continue with 6.

   Case 2: If the gscBsiUtilPassthru() call does not return the code BSI_ INSUFFICIENT_BUFFER, then print
   "gscBsiUtilPassthru() (Discovery Method 2, Discovery Mode) did not return the code BSI_INSUFFICIENT_BUFFER.
   Status: Test 11.2 Failed.
   End Test for Assertion 11.2.

6. Re-allocate the unsigned char* uszCardResponse1102 with length == punCardResponseLen1102.

7. Make a gscBsiUtilPassthru() call to the SPS, using
   - hCard == hCard1100
   - uszCardCommand == _goodCardCommand
   - unCardCommandLen == _goodCardCommandLen
   - uszCardResponse == uszCardResponse1102
   - punCardResponseLen == punCardResponseLen1102.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_OK
   - uszCardResponse1102 == a string containing the APDU response from the connected card.

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiUtilPassthru() call returns the code BSI_OK, then:.

   Case 1.1: If uszCardResponse1102 == one of the elements of _goodCardResponse, then print
   "gscBsiUtilPassthru() (Discovery Method 2, Final Mode) called with valid parameters has been verified.
   Status: Test 11.2 Passed."

   Case 1.2: If uszCardResponse1102 is not one of the elements of _goodCardResponse, then print
   "gscBsiUtilPassthru() (Discovery Method 2, Final Mode) called with valid parameters has not been verified.
   Status: Test 11.2 Failed."

   Case 2: If the gscBsiUtilPassthru() call does not return the code BSI_OK, then print
   "gscBsiUtilPassthru() (Discovery Method 2, Final Mode) called with valid parameters returned an incorrect code.
   Status: Test 11.2 Failed."

Test for Assertion 11.3

The function is tested using a bad handle (Discovery Method 1, Discovery Mode).

Instantiation Scenario:
1. (Pre) Construct the Starting State for the testing of gscBsiUtilPassthru().

2. (Pre) Allocate the unsigned long* punCardResponseLen1103 == 0.

8. (Pre) Print "Testing of Assertion 11.3".

3. Make a gscBsiUtilPassthru() call to the SPS, using
   - hCard /= hCard1100
   - uszCardCommand == _goodCardCommand

- unCardCommandLen == _goodCardCommandLen
- uszCardResponse == NULL
- punCardResponseLen == punCardResponseLen1103.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_BAD_HANDLE.

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiUtilPassthru() call returns the code BSI_BAD_HANDLE, then:

   Perform Test for Assertion 9.11.3.1 using
   - hCard == hCard1100.

   Print
   "gscBsiUtilPassthru() (Discovery Method 1, Discovery Mode) called with a bad handle has been verified.
   Status: Test 11.3 Passed."

   Case 2: If the gscBsiUtilPassthru() call does not return the code BSI_BAD_HANDLE, then print
   "gscBsiUtilPassthru() (Discovery Method 1, Discovery Mode) called with a bad handle returned an incorrect code.
   Status: Test 11.3 Failed."

Test for Assertion 11.4

The function is tested using a bad handle (Discovery Method 1, Final Mode).

Instantiation Scenario:
1. (Pre) Construct the Starting State for the testing of gscBsiUtilPassthru().

2. (Pre) Allocate the unsigned long* punCardResponseLen1104 == 0.

3. (Pre) Print "Testing of Assertion 11.4".

4. Make a gscBsiUtilPassthru() call to the SPS, using
   - hCard == hCard1100
   - uszCardCommand == _goodCardCommand
   - unCardCommandLen == _goodCardCommandLen
   - uszCardResponse == NULL
   - punCardResponseLen == punCardResponseLen1104.

5. Allocate the unsigned char* uszCardResponse1104 with length == punCardResponseLen1104.

6. Make a gscBsiUtilPassthru() call to the SPS, using
   - hCard /= hCard1100
   - uszCardCommand == _goodCardCommand
   - unCardCommandLen == _goodCardCommandLen
   - uszCardResponse == uszCardResponse1104
   - punCardResponseLen == punCardResponseLen1104.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_BAD_HANDLE.

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiUtilPassthru() call returns the code BSI_BAD_HANDLE, then:

   Perform Test for Assertion 9.11.4.1 using
   - hCard == hCard1100.

   Print
   "gscBsiUtilPassthru() (Discovery Method 1, Final Mode) called with a bad handle has been verified.
   Status: Test 11.4 Passed."

   Case 2: If the gscBsiUtilPassthru() call does not return the code BSI_BAD_HANDLE, then print
   "gscBsiUtilPassthru() (Discovery Method 1, Final Mode) called with a bad handle returned an incorrect code.
   Status: Test 11.4 Failed."

Test for Assertion 11.5

The function is tested with another application having established a transaction lock.

Until we encounter implementations that allow multiple simultaneous applications, I don't think we need to worry about this assertion.

Test for Assertion 11.6

The function is tested using a bad CardCommand length (Discovery Method 1, Discovery Mode).

Instantiation Scenario:
1. (Pre) Construct the Starting State for the testing of gscBsiUtilPassthru().

2. (Pre) Allocate the unsigned long* punCardResponseLen1106 == 0.

3. (Pre) Print "Testing of Assertion 11.6".

4. Make a gscBsiUtilPassthru() call to the SPS, using
   - hCard == hCard1100
   - uszCardCommand == _goodCardCommand
   - unCardCommandLen == 1
   - uszCardResponse == NULL
   - punCardResponseLen == punCardResponseLen1106.

Verification Goal:
To verify the Expected Results:
1. The call returns

- the return code BSI_BAD_PARAM.

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiUtilPassthru() call returns the code BSI_BAD_PARAM, then:

   Perform Test for Assertion 9.11.6.1 using
   - hCard == hCard1100.

   Print
   "gscBsiUtilPassthru() (Discovery Method 1, Discovery Mode) called with a bad CardCommand length has been verified.
   Status: Test 11.6 Passed."

   Case 2: If the gscBsiUtilPassthru() call does not return the code BSI_BAD_PARAM, then print
   "gscBsiUtilPassthru() (Discovery Method 1, Discovery Mode) called with a bad CardCommand length returned an incorrect code.
   Status: Test 11.6 Failed."

Test for Assertion 11.7

The function is tested using a bad CardCommand length (Discovery Method 1, Final Mode).

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiUtilPassthru().

2. (Pre) Allocate the unsigned long* punCardResponseLen1107 == 0.

3. (Pre) Print "Testing of Assertion 11.7".

4. Make a gscBsiUtilPassthru() call to the SPS, using
   - hCard == hCard1100
   - uszCardCommand == _goodCardCommand
   - unCardCommandLen == _goodCardCommandLen
   - uszCardResponse == NULL
   - punCardResponseLen == punCardResponseLen1107.

5. Allocate the unsigned char* uszCardResponse1107 with length == punCardResponseLen1107.

6. Make a gscBsiUtilPassthru() call to the SPS, using
   - hCard == hCard1100
   - uszCardCommand == _goodCardCommand
   - unCardCommandLen == 1
   - uszCardResponse == uszCardResponse1107
   - punCardResponseLen == punCardResponseLen1107.

Verification Goal:

To verify the Expected Results:

1. The call returns
   - the return code BSI_BAD_PARAM.

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiUtilPassthru() call returns the code BSI_BAD_PARAM, then:

   Perform Test for Assertion 9.11.7.1 using
   - hCard == hCard1100.

   Print
   "gscBsiUtilPassthru() (Discovery Method 1, Final Mode) called with a bad CardCommand length has been verified.
   Status: Test 11.7 Passed."

   Case 2: If the gscBsiUtilPassthru() call does not return the code BSI_BAD_PARAM, then print
   "gscBsiUtilPassthru() (Discovery Method 1, Final Mode) called with a bad CardCommand length returned an incorrect code.
   Status: Test 11.7 Failed."

Test for Assertion 11.8

The function is tested using a bad CardResponse length (Discovery Method 1).

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiUtilPassthru().

2. (Pre) Allocate the unsigned long* punCardResponseLen1108 == 0.

3. (Pre) Print "Testing of Assertion 11.8".

4. Make a gscBsiUtilPassthru() call to the SPS, using
   - hCard == hCard1100
   - uszCardCommand == _goodCardCommand
   - unCardCommandLen == _goodCardCommandLen
   - uszCardResponse == NULL
   - punCardResponseLen == punCardResponseLen1108.

5. Allocate the unsigned char* uszCardResponse1108 with length == punCardResponseLen1108.

6. Assign punCardResponseLen1108 == 0.

7. Make a gscBsiUtilPassthru() call to the SPS, using
   - hCard == hCard1100
   - uszCardCommand == _goodCardCommand
   - unCardCommandLen == _goodCardCommandLen
   - uszCardResponse == uszCardResponse1108
   - punCardResponseLen == punCardResponseLen1108.

Verification Goal:

To verify the Expected Results:

1. The call returns
   - the return code BSI_BAD_PARAM.

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiUtilPassthru() call returns the code BSI_BAD_PARAM, then:

   Perform Test for Assertion 9.11.8.1 using
   - hCard == hCard1100.

   Print
   "gscBsiUtilPassthru() (Discovery Method 2, Final Mode) called with a bad CardResponse length has been verified.
   Status: Test 11.8 Passed."

   Case 2: If the gscBsiUtilPassthru() call does not return the code BSI_BAD_PARAM, then print
   "gscBsiUtilPassthru() (Discovery Method 1, Final Mode) called with a bad CardResponse length returned an incorrect code.
   Status: Test 11.8 Failed."

Test for Assertion 11.9

The function is tested with a removed card (Discovery Method 1, Discovery Mode).

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiUtilPassthru().

2. (Pre) Allocate the unsigned long* punCardResponseLen1109 == 0.

3. (Pre) Print "Testing of Assertion 11.9".

4. Remove the connected card from the reader.

5. Make a gscBsiUtilPassthru() call to the SPS, using
   - hCard == hCard1100
   - uszCardCommand == _goodCardCommand
   - unCardCommandLen == _goodCardCommandLen
   - uszCardResponse == NULL
   - punCardResponseLen == punCardResponseLen1109.

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_CARD_REMOVED.

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiUtilPassthru() call returns the code BSI_CARD_REMOVED, then:

   Print
   "gscBsiUtilPassthru() (Discovery Method 1, Discovery Mode) called with the connected card removed has been verified.
   Status: Test 11.9 Passed."

   Case 2: If the gscBsiUtilPassthru() call does not return the code BSI_CARD_REMOVED, then print

"gscBsiUtilPassthru() (Discovery Method 1, Discovery Mode) called with the connected card removed returned an incorrect code.
Status: Test 11.9 Failed."

Test for Assertion 11.10

The function is tested with a removed card (Discovery Method 1, Final Mode).

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiUtilPassthru().

2. (Pre) Allocate the unsigned long* punCardResponseLen1110 == 0.

3. (Pre) Print "Testing of Assertion 11.10".

4. Make a gscBsiUtilPassthru() call to the SPS, using
   - hCard == hCard1100
   - uszCardCommand == _goodCardCommand
   - unCardCommandLen == _goodCardCommandLen
   - uszCardResponse == NULL
   - punCardResponseLen == punCardResponseLen1110.

5. Allocate the unsigned char* uszCardResponse1110 with length == punCardResponseLen1110.

6. Remove the connected card from the reader.

7. Make a gscBsiUtilPassthru() call to the SPS, using
   - hCard == hCard1100
   - uszCardCommand == _goodCardCommand
   - unCardCommandLen == _goodCardCommandLen
   - uszCardResponse == uszCardResponse1110
   - punCardResponseLen == punCardResponseLen1110.

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_CARD_REMOVED.

Verification and Reporting Scenario:

Case 1: If the gscBsiUtilPassthru() call returns the code BSI_CARD_REMOVED, then:

Print
"gscBsiUtilPassthru() (Discovery Method 1, Final Mode) called with the connected card removed has been verified.
Status: Test 11.10 Passed."

Case 2: If the gscBsiUtilPassthru() call does not return the code BSI_CARD_REMOVED, then print
"gscBsiUtilPassthru() (Discovery Method 1, Final Mode) called with the connected card removed returned an incorrect code.
Status: Test 11.10 Failed."

12. gscBsiUtilReleaseContext()

Starting State for Each Test:

1. A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1200.

2. There exists a container on the connected card for which
   - readTagListACR is PIN Protected
   - the value of the PIN is _PIN
   - the container is represented by AID value ==
     - _goodGSCAID1, of length _goodGSCAID1Len (GSC), or
     - _goodCACAID1, of length _goodCACAID1Len (CAC).

3. There does not exist a container on the card with AID value == _badGSCAID (GSC) or _badCACAID (CAC).

4. There exists a BSIAuthenticator array strctAuthenticator1200 with one element, the structure BSIAuthenticator1200. This structure has fields
   - unAccessMethodType == BSI_AM_PIN
   - unkeyIDOrReference == _keyIDOrReference1
   - uszAuthValue == _goodAuthValue1
   - unAuthValueLen == _goodAuthValue1Len.

5. There exists an unsigned long* punNbTags1200 == 0.

6. An authenticated session is established with the target container:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard1200
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - strctAuthenticator == strctAuthenticator1200
   - unAuthNb == 1.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with step 2 of the Instantiation Scenario.

   Case 2: If the gscBsiUtilGetCardStatus() call does not return the code BSI_OK, then print
   "A session cannot be established. gscBsiUtilReleaseContext() cannot be tested".
   End current test.

Test for Assertion 12.1

The function is tested with valid parameters.

Instantiation Scenario:

1. (Pre) Print "Testing of Assertion 12.1".

2. (Pre) Construct the Starting State for the testing of gscBsiUtilReleaseContext().

3. Make a gscBsiUtilReleaseContext() call to the SPS, using
   - hCard == hCard1200
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC).

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_OK.

2. There is no longer an authenticated session established with the target container.

Perform this verification by issuing a call to gscBsiGcReadTagList().

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiUtilReleaseContext() call returns the code BSI_OK, then verify that there is no longer an authenticated session established with the target container:

   Make a gscBsiGcReadTagList() call to the SPS, using
   - hCard == hCard1200
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - tagArray == NULL
   - punNbTags == punNbTags1200.

   Case 1.1: If the gscBsiGcReadTagList() call does not return the code BSI_OK, then print
   "gscBsiUtilReleaseContext() called with valid parameters has been verified because a subsequent call to gscBsiGcReadTagList() was unsuccessful, indicating that the authenticated session had been terminated.
   Status: Test 12.1 Passed."

   Case 1.2: If the gscBsiGcReadTagList() call returns the code BSI_OK, then print
   "gscBsiUtilReleaseContext() called with valid parameters has not been verified because a subsequent call to gscBsiGcReadTagList() was successful, indicating that the authenticated session had not been terminated.
   Status: Test 12.1 Failed."

   Case 2: If the gscBsiUtilReleaseContext() call does not return the code BSI_OK, then print
   "gscBsiUtilReleaseContext() called with valid parameters returned an incorrect code.
   Status: Test 12.1 Failed."

Test for Assertion 12.2

The function is tested using a bad handle.

Instantiation Scenario:

1. (Pre) Print "Testing of Assertion 12.2".

2. (Pre) Construct the Starting State for the testing of gscBsiUtilReleaseContext().

3. Make a gscBsiUtilReleaseContext() call to the SPS, using
   - hCard /= hCard1200
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC).

Verification Goal:

To verify the Expected Results:

1. The call returns
   - the return code BSI_BAD_HANDLE.

2. There continues to be an authenticated session established with the target container.

Perform this verification by issuing a call to gscBsiGcReadTagList().

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiUtilReleaseContext() call returns the code BSI_BAD_HANDLE, then:

   Perform Test for Assertion 9.12.2.1 using
   - hCard == hCard1200.

   Verify that there continues to be an authenticated session established with the target container:

   Make a gscBsiGcReadTagList() call to the SPS, using
   - hCard == hCard1200
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - tagArray == NULL
   - punNbTags == punNbTags1200.

   Case 1.1: If the gscBsiGcReadTagList() call returns the code BSI_OK, then print
   "gscBsiUtilReleaseContext() called with a bad handle has been verified because a subsequent call to gscBsiGcReadTagList() was successful, indicating that the authenticated session had not been terminated.
   Status: Test 12.2 Passed."

   Case 1.2: If the gscBsiGcReadTagList() call does not return the code BSI_OK, then print
   "gscBsiUtilReleaseContext() called with a bad handle has not been verified because a subsequent call to gscBsiGcReadTagList() was successful, indicating that the authenticated session had been terminated.
   Status: Test 12.2 Failed."

Case 2: If the gscBsiUtilReleaseContext() call does not return the code BSI_BAD_HANDLE, then print
"gscBsiUtilReleaseContext() called with a bad handle returned an incorrect code.
Status: Test 12.2 Failed."

Test for Assertion 12.3

The function is tested using a bad AID value.

Instantiation Scenario:

1. (Pre) Print "Testing of Assertion 12.3".

2. (Pre) Construct the Starting State for the testing of gscBsiUtilReleaseContext().

3. Make a gscBsiUtilReleaseContext() call to the SPS, using
   - hCard == hCard1200
   - uszAID == _badGSCAID (GSC) or _badCACAID (CAC)
   - unAIDLen == _badGSCAIDLen (GSC) or _badCACAIDLen (CAC).

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_BAD_AID.

2. There continues to be an authenticated session established with the target container.

Perform this verification by issuing a call to gscBsiGcReadTagList().

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiUtilReleaseContext() call returns the code BSI_BAD_AID, then:

   Perform Test for Assertion 9.12.3.1 using
   - hCard == hCard1200.

   Verify that there continues to be an authenticated session established with the target container:

   Make a gscBsiGcReadTagList() call to the SPS, using
   - hCard == hCard1200
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - tagArray == NULL
   - punNbTags == punNbTags1200.

   Case 1.1: If the gscBsiGcReadTagList() call returns the code BSI_OK, then print
   "gscBsiUtilReleaseContext() called with a bad AID value has been verified because a subsequent call to gscBsiGcReadTagList() was successful, indicating that the authenticated session had not been terminated.
   Status: Test 12.3 Passed."

Case 1.2: If the gscBsiGcReadTagList() call does not return the code BSI_OK, then print
"gscBsiUtilReleaseContext() called with a bad AID value has not been verified because a subsequent call to gscBsiGcReadTagList() was unsuccessful, indicating that the authenticated session had been terminated.
Status: Test 12.3 Failed."

Case 2: If the gscBsiUtilReleaseContext() call does not return the code BSI_BAD_AID, then print
"gscBsiUtilReleaseContext() called with a bad AID value returned an incorrect code.
Status: Test 12.3 Failed."

Test for Assertion 12.4

The function is tested using a bad AID length.

Instantiation Scenario:

1. (Pre) Print "Testing of Assertion 12.4".

2. (Pre) Construct the Starting State for the testing of gscBsiUtilReleaseContext().

3. Make a gscBsiUtilReleaseContext() call to the SPS, using
   - hCard == hCard1200
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _badAIDLen.

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_BAD_PARAM.

2. There continues to be an authenticated session established with the target container.

Perform this verification by issuing a call to gscBsiGcReadTagList().

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiUtilReleaseContext() call returns the code BSI_BAD_PARAM, then:

   Perform Test for Assertion 9.12.4.1 using
   - hCard == hCard1200.

   Verify that there continues to be an authenticated session established with the target container:

   Make a gscBsiGcReadTagList() call to the SPS, using
   - hCard == hCard1200
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - tagArray == NULL

- punNbTags == punNbTags1200.

Case 1.1: If the gscBsiGcReadTagList() call returns the code BSI_OK, then print
"gscBsiUtilReleaseContext() called with a bad AID length has been verified because a subsequent call to gscBsiGcReadTagList() was successful, indicating that the authenticated session had not been terminated.
Status: Test 12.4 Passed."

Case 1.2: If the gscBsiGcReadTagList() call does not return the code BSI_OK, then print
"gscBsiUtilReleaseContext() called with a bad AID length has not been verified because a subsequent call to gscBsiGcReadTagList() was unsuccessful, indicating that the authenticated session had been terminated.
Status: Test 12.4 Failed."

Case 2: If the gscBsiUtilReleaseContext() call does not return the code BSI_BAD_PARAM, then print
"gscBsiUtilReleaseContext() called with a bad AID length returned an incorrect code.
Status: Test 12.4 Failed."

Test for Assertion 12.5

The function is tested with a card that has been removed.

Instantiation Scenario:

1. (Pre) Print "Testing of Assertion 12.5".

2. (Pre) Construct the Starting State for the testing of gscBsiUtilReleaseContext().

3. Remove the connected card from the reader.

4. Make a gscBsiUtilReleaseContext() call to the SPS, using
   - hCard == hCard1200
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC).

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_CARD_REMOVED.

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiUtilReleaseContext() call returns the code BSI_CARD_REMOVED, then print
   "gscBsiUtilReleaseContext() called with the connected card removed has been verified.
   Status: Test 12.5 Passed."

   Case 2: If the gscBsiUtilReleaseContext() call does not return the code BSI_CARD_REMOVED, then print

"gscBsiUtilReleaseContext() called with the connected card removed returned an incorrect code.
Status: Test 12.5 Failed."

Test for Assertion 12.6

The function is tested with another application having established a transaction lock.

Until we encounter implementations that allow multiple simultaneous applications, I don't think we need to worry about this assertion.

13. gscBsiGcDataCreate()

Starting State for Each Test:

1. There exists a BSIAuthenticator array strctAuthenticator1300 with one element, the structure BSIAuthenticator1300. This structure has fields
   - unAccessMethodType == BSI_AM_PIN
   - unkeyIDOrReference == _keyIDOrReference1
   - uszAuthValue == _goodAuthValue1
   - unAuthValueLen == _goodAuthValue1Len.

Test for Assertion 13.1

The function is tested using valid parameters.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGcDataCreate().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1301.

3. (Pre) There exists a target container on the connected card with the following properties:
   - the ACR for the gscBsiGcDataCreate() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by AID value ==
     - _goodGSCAID1, of length _goodGSCAID1Len (GSC), or
   - _goodCACAID1, of length _goodCACAID1Len (CAC)
   - there does not exist a data item on the target container with tag == _newTag1301
   - the container can accommodate _newDvalue1301.

4. (Pre) Print "Testing of Assertion 13.1".

5. (Pre) Establish an authenticated session with the target container on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard1301
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - strctAuthenticator == strctAuthenticator1300
   - unAuthNb == 1.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 6.

   Case 2: If the gscBsiUtilAcquireContext() call does not return the code BSI_OK, then print
   "A session cannot be established. Assertion 13.1 of gscBsiGcDataCreate() cannot be tested".
   End Test for Assertion 13.1.

6. Make a gscBsiGcDataCreate() call to the SPS, using
    - hCard == hCard1301
    - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
    - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
    - ucTag == _newTag1301
    - uszValue == _newDvalue1301
    - unValueLen == _newDvalueLen.

Verification Goal:
To verify the Expected Results:

1. The call returns
    - the return code BSI_OK or the return code BSI_NO_CARDSERVICE.

2. If the return code is BSI_OK, then the data value _newDvalue1301 is stored, with tag == _newTag1301, in the target container.

3. No other changes are made to the container structure of the connected card.

Note: I suppose if we were very ambitious, we would verify 3., by comparing before and after snapshots of every container on the connected card. For now, I propose limiting ourselves to verifying that the specified data value is correctly stored in the selected container.

Perform this verification by issuing a call to gscBsiGcReadValue().

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiGcDataCreate() call returns the code BSI_OK, then verify that that the specified data value has indeed been stored, with the specified tag, in the target container.

    Allocate the unsigned char* uszValue1301 with length _newDvalueLen.

    Make a gscBsiGcReadValue() call to the SPS, using
    - hCard == hCard1301
    - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
    - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
    - ucTag == _newTag1301
    - uszValue == uszValue1301
    - punValueLen == _newDvalueLen.

    Case 1.1: If the gscBsiGcReadValue() call returns
    - the code BSI_OK
    - uszValue1301 == _newDvalue1301

    then print
    "gscBsiGcDataCreate() called with valid parameters has been verified because a subsequent call to gscBsiGcReadValue() was successful, indicating that the specified data value was correctly created.
    Status: Test 13.1 Passed."

    Case 1.2: If the gscBsiReadValue() call returns
    - the code BSI_OK
    - uszValue1301 /= _newDvalue1301

then print
"gscBsiGcDataCreate() called with valid parameters has not been verified because a subsequent call to gscBsiGcReadValue() indicated that the specified data value was not correctly created.
Status: Test 13.1 Failed."

Case 1.3: If the gscBsiGcReadValue() call does not return the code BSI_OK, then print
"gscBsiGcDataCreate() called with valid parameters has not been verified because a subsequent call to gscBsiGcReadValue() was ambiguous.
Status: Test 13.1 Undetermined."

Case 2: If the gscBsiGcDataCreate() call returns the code BSI_NO_CARDSERVICE, then print
"gscBsiGcDataCreate() is not supported.
Status: Test 13.1 Not Supported."

Case 3: If the gscBsiGcDataCreate() call does not return the code BSI_OK or the code BSI_NO_CARDSERVICE, then print
"gscBsiGcDataCreate() called with valid parameters returned an incorrect code.
Status: Test 13.1 Failed."

Test for Assertion 13.2

The function is tested using a bad handle.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGcDataCreate().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1302.

3. (Pre) There exists a target container on the connected card with the following properties:
   - the ACR for the gscBsiGcDataCreate() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by AID value ==
     - _goodGSCAID1, of length _goodGSCAID1Len (GSC), or
   - _goodCACAID1, of length _goodCACAID1Len (CAC)
   - there does not exist a data item on the target container with tag == _newTag1302
   - the target container can accommodate _newDvalue1302.

4. (Pre) Print "Testing of Assertion 13.2".

5. (Pre) Establish an authenticated session with the target container on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard1302
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)

- unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
- strctAuthenticator == strctAuthenticator1300
- unAuthNb == 1.

Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 6.

Case 2: If the gscBsiUtilAcquireContext() call does not return the code BSI_OK, then print
"A session cannot be established. Assertion 13.2 of gscBsiGcDataCreate() cannot be tested".
End Test for Assertion 13.2.

6. Make a gscBsiGcDataCreate() call to the SPS, using
   - hCard /= hCard1302
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - ucTag == _newTag1302
   - uszValue == _newDvalue1302
   - unValueLen == _newDvalueLen.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_BAD_HANDLE or the return code BSI_NO_CARDSERVICE.

2. No changes are made to the container structure of the connected card.

Perform this verification by issuing a call to gscBsiGcReadValue().

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiGcDataCreate() call returns the code BSI_BAD_HANDLE, then

   Perform Test for Assertion 9.13.2.1 using
   - hCard == hCard1302.

   Verify that that the specified data value was not stored, with the specified tag, in the target container.

   Allocate the unsigned char* uszValue1302 with length _newDvalueLen.

   Make a gscBsiGcReadValue() call to the SPS, using
   - hCard == hCard1302
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - ucTag == _newTag1302
   - uszValue == uszValue1302
   - punValueLen == _newDvalueLen.

   Case 1.1: If the gscBsiGcReadValue() call returns either of the codes:

- BSI_BAD_TAG
- BSI_IO_ERROR

then print
"gscBsiGcDataCreate() called with a bad handle has been verified because a subsequent call to gscBsiGcReadValue() did not find the data item.
Status: Test 13.2 Passed."

Case 1.2: If the gscBsiReadValue() call returns the code BSI_OK, then print
"gscBsiGcDataCreate() called with a bad handle has not been been verified because a subsequent call to gscBsiGcReadValue() indicated that the data value had been created.
Status: Test 13.2 Failed."

Case 1.3: If the gscBsiGcReadValue() call does not return any of the codes
- BSI_BAD_TAG
- BSI_IO_ERROR
- BSI_OK

then print
"gscBsiGcDataCreate() called with a bad handle has not been verified because a subsequent call to gscBsiGcReadValue() was ambiguous.
Status: Test 13.2 Undetermined."

Case 2: If the gscBsiGcDataCreate() call returns the code BSI_NO_CARDSERVICE, then print
"gscBsiGcDataCreate() is not supported.
Status: Test 13.2 Not Supported."

Case 3: If the gscBsiGcDataCreate() call does not return the code BSI_BAD_HANDLE or the code BSI_NO_CARDSERVICE, then print
"gscBsiGcDataCreate() called with a bad handle returned an incorrect code.
Status: Test 13.2 Failed."

Test for Assertion 13.3

The function is tested using a bad AID value.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGcDataCreate().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1303.

3. (Pre) There exists a target container on the connected card with the following properties:
   - the ACR for the gscBsiGcDataCreate() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by AID value ==
     - _goodGSCAID1, of length _goodGSCAID1Len (GSC), or

- _goodCACAID1, of length _goodCACAID1Len (CAC)
- there does not exist a data item on the target container with tag == _newTag1303
- the target container can accommodate _newDvalue1303.

4. (Pre) There does not exist a container on the card with AID value == _badGSCAID (GSC) or _badCACAID (CAC).

5. (Pre) Print "Testing of Assertion 13.3".

6. (Pre) Establish an authenticated session with the target container on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard1303
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - strctAuthenticator == strctAuthenticator1300
   - unAuthNb == 1.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 7.

   Case 2: If the gscBsiUtilAcquireContext() call does not return the code BSI_OK, then print
   "A session cannot be established. Assertion 13.3 of gscBsiGcDataCreate() cannot be tested".
   End Test for Assertion 13.3.

7. Make a gscBsiGcDataCreate() call to the SPS, using
   - hCard == hCard1303
   - uszAID == _badGSCAID (GSC) or _badCACAID (CAC)
   - unAIDLen == _badGSCAIDLen (GSC) or _badCACAIDLen (CAC)
   - ucTag == _newTag1303
   - uszValue == _newDvalue1303
   - unValueLen == _newDvalueLen.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_BAD_AID or the return code BSI_NO_CARDSERVICE.

2. No changes are made to the container structure of the connected card.

Perform this verification by issuing a call to gscBsiGcReadValue().

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiGcDataCreate() call returns the code BSI_BAD_AID, then:

   Perform Test for Assertion 9.13.3.1 using
   - hCard == hCard1303.

Verify that that the specified data value was not stored, with the specified tag, in the target container.

Allocate the unsigned char* uszValue1303 with length _newDvalueLen.

Make a gscBsiGcReadValue() call to the SPS, using
- hCard == hCard1303
- uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
- unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
- ucTag == _newTag1303
- uszValue == uszValue1303
- punValueLen == _newDvalueLen.

Case 1.1: If the gscBsiGcReadValue() call returns either of the codes:
- BSI_BAD_TAG
- BSI_IO_ERROR

then print
"gscBsiGcDataCreate() called with a bad AID value has been verified because a subsequent call to gscBsiGcReadValue() did not find the data item.
Status: Test 13.3 Passed."

Case 1.2: If the gscBsiReadValue() call returns the code BSI_OK, then print
"gscBsiGcDataCreate() called with a bad AID value has not been been verified because a subsequent call to gscBsiGcReadValue() indicated that the data value had been created.
Status: Test 13.3 Failed."

Case 1.3: If the gscBsiGcReadValue() call does not return any of the codes
- BSI_BAD_TAG
- BSI_IO_ERROR
- BSI_OK

then print
"gscBsiGcDataCreate() called with a bad AID value has not been verified because a subsequent call to gscBsiGcReadValue() was ambiguous.
Status: Test 13.3 Undetermined."

Case 2: If the gscBsiGcDataCreate() call returns the code BSI_NO_CARDSERVICE, then print
"gscBsiGcDataCreate() is not supported.
Status: Test 13.3 Not Supported."

Case 3: If the gscBsiGcDataCreate() call does not return the code BSI_BAD_AID or the code BSI_NO_CARDSERVICE, then print
"gscBsiGcDataCreate() called with a bad AID value returned an incorrect code.
Status: Test 13.3 Failed."

Test for Assertion 13.4

The function is tested with another application having established a transaction lock.

Until we encounter implementations that allow multiple simultaneous applications, I don't think we need to worry about this assertion.

Test for Assertion 13.5

The function is tested using a bad AID length.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGcDataCreate().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1305.

3. (Pre) There exists a target container on the connected card with the following properties:
   - the ACR for the gscBsiGcDataCreate() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by AID value ==
     - _goodGSCAID1, of length _goodGSCAID1Len (GSC), or
   - _goodCACAID1, of length _goodCACAID1Len (CAC)
   - there does not exist a data item on the target container with tag == _newTag1305
   - the target container can accommodate _newDvalue1305.

4. (Pre) Print "Testing of Assertion 13.5".

5. (Pre) Establish an authenticated session with the target container on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard1305
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - strctAuthenticator == strctAuthenticator1300
   - unAuthNb == 1.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 6.

   Case 2: If the gscBsiUtilAcquireContext() call does not return the code BSI_OK, then print
   "A session cannot be established. Assertion 13.5 of gscBsiGcDataCreate() cannot be tested".
   End Test for Assertion 13.5.

6. Make a gscBsiGcDataCreate() call to the SPS, using
   - hCard == hCard1305
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _badAIDLen

- ucTag == _newTag1305
- uszValue == _newDvalue1305
- unValueLen == _newDvalueLen.

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_BAD_PARAM or the return code BSI_NO_CARDSERVICE.

2. No changes are made to the container structure of the connected card.

Perform this verification by issuing a call to gscBsiGcReadValue().

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiGcDataCreate() call returns the code BSI_BAD_PARAM, then

   Perform Test for Assertion 9.13.5.1 using
   - hCard == hCard1305.

   Verify that that the specified data value was not stored, with the specified tag, in the target container.

   Allocate the unsigned char* uszValue1305 with length _newDvalueLen.

   Make a gscBsiGcReadValue() call to the SPS, using
   - hCard == hCard1305
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - ucTag == _newTag1305
   - uszValue == uszValue1305
   - punValueLen == _newDvalueLen.

   Case 1.1: If the gscBsiGcReadValue() call returns either of the codes:
   - BSI_BAD_TAG
   - BSI_IO_ERROR

   then print
   "gscBsiGcDataCreate() called with a bad AID length has been verified because a subsequent call to gscBsiGcReadValue() did not find the data item.
   Status: Test 13.5 Passed."

   Case 1.2: If the gscBsiReadValue() call returns the code BSI_OK, then print
   "gscBsiGcDataCreate() called with a bad AID length has not been been verified because a subsequent call to gscBsiGcReadValue() indicated that the data value had been created.
   Status: Test 13.5 Failed."

   Case 1.3: If the gscBsiGcReadValue() call does not return any of the codes

- BSI_BAD_TAG
- BSI_IO_ERROR
- BSI_OK

then print
"gscBsiGcDataCreate() called with a bad AID length has not been verified because a subsequent call to gscBsiGcReadValue() was ambiguous.
Status: Test 13.5 Undetermined."

Case 2: If the gscBsiGcDataCreate() call returns the code BSI_NO_CARDSERVICE, then print
"gscBsiGcDataCreate() is not supported.
Status: Test 13.5 Not Supported."

Case 3: If the gscBsiGcDataCreate() call does not return the code BSI_BAD_PARAM or the code BSI_NO_CARDSERVICE, then print
"gscBsiGcDataCreate() called with a bad AID length returned an incorrect code.
Status: Test 13.5 Failed."

Test for Assertion 13.6

The function is tested using a bad data value length.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGcDataCreate().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1306.

3. (Pre) There exists a target container on the connected card with the following properties:
   - the ACR for the gscBsiGcDataCreate() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by AID value ==
     - _goodGSCAID1, of length _goodGSCAID1Len (GSC), or
   - _goodCACAID1, of length _goodCACAID1Len (CAC)
   - there does not exist a data item on the target container with tag == _newTag1306
   - the target container can accommodate _newDvalue1306.

4. (Pre) Print "Testing of Assertion 13.6".

5. (Pre) Establish an authenticated session with the target container on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard1306
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - strctAuthenticator == strctAuthenticator1300
   - unAuthNb == 1.

Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 6.

Case 2: If the gscBsiUtilAcquireContext() call does not return the code BSI_OK, then print
"A session cannot be established. Assertion 13.6 of gscBsiGcDataCreate() cannot be tested".

End Test for Assertion 13.6.

6. (Pre) Define the unsigned long unValueLen1306 == 0.

7. Make a gscBsiGcDataCreate() call to the SPS, using
   - hCard == hCard1306
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - ucTag == _newTag1306
   - uszValue == _newDvalue1306
   - unValueLen == _badDvalueLen.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_BAD_PARAM or the return code BSI_NO_CARDSERVICE.

2. No changes are made to the container structure of the connected card.

Perform this verification by issuing a call to gscBsiGcReadValue().

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiGcDataCreate() call returns the code BSI_BAD_PARAM, then

   Perform Test for Assertion 9.13.6.1 using
   - hCard == hCard1306.

   Verify that that the specified data value was not stored, with the specified tag, in the target container.

   Allocate the unsigned char* uszValue1306 with length _newDvalueLen.

   Make a gscBsiGcReadValue() call to the SPS, using
   - hCard == hCard1306
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - ucTag == _newTag1306
   - uszValue == uszValue1306
   - punValueLen == _newDvalueLen.

   Case 1.1: If the gscBsiGcReadValue() call returns either of the codes:
   - BSI_BAD_TAG
   - BSI_IO_ERROR

   then print

"gscBsiGcDataCreate() called with a bad data value length has been verified because a subsequent call to gscBsiGcReadValue() did not find the data item.
Status: Test 13.6 Passed."

Case 1.2: If the gscBsiReadValue() call returns the code BSI_OK, then print
"gscBsiGcDataCreate() called with a bad data value length has not been been verified because a subsequent call to gscBsiGcReadValue() indicated that the data value had been created.
Status: Test 13.6 Failed."

Case 1.3: If the gscBsiGcReadValue() call does not return any of the codes
- BSI_BAD_TAG
- BSI_IO_ERROR
- BSI_OK

then print
"gscBsiGcDataCreate() called with a bad data value length has not been verified because a subsequent call to gscBsiGcReadValue() was ambiguous.
Status: Test 13.6 Undetermined."

Case 2: If the gscBsiGcDataCreate() call returns the code BSI_NO_CARDSERVICE, then print
"gscBsiGcDataCreate() is not supported.
Status: Test 13.6 Not Supported."

Case 3: If the gscBsiGcDataCreate() call does not return the code BSI_BAD_PARAM or the code BSI_NO_CARDSERVICE, then print
"gscBsiGcDataCreate() called with a bad data value length returned an incorrect code.
Status: Test 13.6 Failed."

Test for Assertion 13.7

The function is tested with a card that has been removed.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGcDataCreate().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1307.

3. (Pre) There exists a target container on the connected card with the following properties:
   - the ACR for the gscBsiGcDataCreate() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by AID value ==
     - _goodGSCAID1, of length _goodGSCAID1Len (GSC), or
   - _goodCACAID1, of length _goodCACAID1Len (CAC)
   - there does not exist a data item on the target container with tag == _newTag1307

- the target container can accommodate _newDvalue1307.

4. (Pre) Print "Testing of Assertion 13.7".

5. (Pre) Establish an authenticated session with the target container on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard1307
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - strctAuthenticator == strctAuthenticator1300
   - unAuthNb == 1.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 6.

   Case 2: If the gscBsiUtilAcquireContext() call does not return the code BSI_OK, then print
   "A session cannot be established. Assertion 13.7 of gscBsiGcDataCreate() cannot be tested".
   End Test for Assertion 13.7.

6. Remove the connected card from the reader.

7. Make a gscBsiGcDataCreate() call to the SPS, using
   - hCard == hCard1307
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - ucTag == _newTag1307
   - uszValue == _newDvalue1307
   - unValueLen == _newDvalueLen.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_CARD_REMOVED or the return code BSI_NO_CARDSERVICE.

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiGcDataCreate() call returns the code BSI_CARD_REMOVED, then print
   "gscBsiGcDataCreate() called with the connected card removed has been verified.
   Status: Test 13.7 Passed."

   Case 2: If the gscBsiGcDataCreate() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiGcDataCreate() is not supported.
   Status: Test 13.7 Not Supported."

   Case 3: If the gscBsiGcDataCreate() call does not return the code BSI_CARD_REMOVED or the code BSI_NO_CARDSERVICE, then print
   "gscBsiGcDataCreate() called with the connected card removed returned an incorrect code.
   Status: Test 13.7 Failed."

Test for Assertion 13.8

The function is tested without fulfilling the applicable ACR.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGcDataCreate().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1308.

3. (Pre) There exists a target container on the connected card with the following properties:
   - the ACR for the gscBsiGcDataCreate() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by AID value ==
     - _goodGSCAID1, of length _goodGSCAID1Len (GSC), or
   - _goodCACAID1, of length _goodCACAID1Len (CAC)
   - the container can accommodate data item _newDvalue1308.

4. (Pre) There does not exist a data item on the target container with tag == _newTag1308.

5. (Pre) Print "Testing of Assertion 13.8".

6. (Pre) Ensure that there is no authenticated session with the target container:

   Make a gscBsiUtilReleaseContext() call to the SPS, using
   - hCard == hCard1308
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC).

7. Make a gscBsiGcDataCreate() call to the SPS, using
   - hCard == hCard1308
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - ucTag == _newTag1308
   - uszValue == _newDvalue1308
   - unValueLen == _newDvalueLen.

Verification Goal:

To verify the Expected Results:

1. The call returns
   - the return code BSI_ACCESS_DENIED or the return code BSI_NO_CARDSERVICE.

2. No changes are made to the container structure of the connected card.

Perform this verification by issuing a call to gscBsiGcReadValue().

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiGcDataCreate() call returns the code BSI_ACCESS_DENIED, then

   Perform Test for Assertion 9.13.8.1 using
   - hCard == hCard1308.

   Verify that that the specified data value was not stored, with the specified tag, in the target container:

   Establish an authenticated session with the target container:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard1308
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - strctAuthenticator == strctAuthenticator1300
   - unAuthNb == 1.

   Case 1.1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then:

   Allocate the unsigned char* uszValue1308 with length _newDvalueLen.

   Make a gscBsiGcReadValue() call to the SPS, using
   - hCard == hCard1308
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - ucTag == _newTag1308
   - uszValue == uszValue1308
   - punValueLen == _newDvalueLen.

   Case 1.1.1: If the gscBsiGcReadValue() call returns either of the codes:
   - BSI_BAD_TAG
   - BSI_IO_ERROR

   then print
   "gscBsiGcDataCreate() called without fulfilling the applicable ACR has been verified because a subsequent call to gscBsiGcReadValue() did not find the data item.
   Status: Test 13.8 Passed."

   Case 1.1.2: If the gscBsiReadValue() call returns the code BSI_OK, then print
   "gscBsiGcDataCreate() called without fulfilling the applicable ACR has not been been verified because a subsequent call to gscBsiGcReadValue() indicated that the data value had been created.
   Status: Test 13.8 Failed."

   Case 1.1.3: If the gscBsiGcReadValue() call does not return any of the codes
   - BSI_OK
   - BSI_BAD_TAG

- BSI_IO_ERROR

then print
"gscBsiGcDataCreate() called without fulfilling the applicable ACR has not been verified because a subsequent call to gscBsiGcReadValue() was ambiguous.
Status: Test 13.8 Undetermined."

Case 1.2: If the gscBsiUtilAcquireContext() call does not return the code BSI_OK, then print
"A session cannot be established. Assertion 13.8 of gscBsiGcDataCreate() cannot be tested".

Case 2: If the gscBsiGcDataCreate() call returns the code BSI_NO_CARDSERVICE, then print
"gscBsiGcDataCreate() is not supported.
Status: Test 13.8 Not Supported."

Case 3: If the gscBsiGcDataCreate() call does not return the code BSI_ACCESS_DENIED or the code BSI_NO_CARDSERVICE, then print
"gscBsiGcDataCreate() called without fulfilling the applicable ACR returned an incorrect code.
Status: Test 13.8 Failed."

Test for Assertion 13.9

The function is tested using a too-large data value.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGcDataCreate().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1309.

3. (Pre) There exists a target container on the connected card with the following properties:
   - the ACR for the gscBsiGcDataCreate() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by AID value ==
     - _goodGSCAID2, of length _goodGSCAID2Len (GSC), or
   - _goodCACAID2, of length _goodCACAID2Len (CAC)
   - the target container contains one data item, which comprises the entire available space of the container. The tag for this data item is _existingTagFull.

4. (Pre) Print "Testing of Assertion 13.9".

5. (Pre) Establish an authenticated session with the target container on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard1309
   - uszAID == _goodGSCAID2 (GSC) or _goodCACAID2 (CAC)
   - unAIDLen == _goodGSCAID2Len (GSC) or _goodCACAID2Len (CAC)

- strctAuthenticator == strctAuthenticator1300
- unAuthNb == 1.

Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 6.

Case 2: If the gscBsiUtilAcquireContext() call does not return the code BSI_OK, then print
"A session cannot be established. Assertion 13.9 of gscBsiGcDataCreate() cannot be tested".
End Test for Assertion 13.9.

6. Make a gscBsiGcDataCreate() call to the SPS, using
   - hCard == hCard1309
   - uszAID == _goodGSCAID2 (GSC) or _goodCACAID2 (CAC)
   - unAIDLen == _goodGSCAID2Len (GSC) or _goodCACAID2Len (CAC)
   - ucTag == _newTag1309
   - uszValue == _newDvalue1309
   - unValueLen == _newDvalueLen.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_NO_MORE_SPACE or the return code BSI_NO_CARDSERVICE.

2. No changes are made to the container structure of the connected card.

Perform this verification by issuing a call to gscBsiGcReadValue().

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiGcDataCreate() call returns the code BSI_NO_MORE_SPACE, then

   Perform Test for Assertion 9.13.9.1 using
   - hCard == hCard1309.

   Verify that that the specified data value was not stored, with the specified tag, in the target container.

   Allocate the unsigned char* uszValue1309 with length _newDvalueLen.

   Make a gscBsiGcReadValue() call to the SPS, using
   - hCard == hCard1309
   - uszAID == _goodGSCAID2 (GSC) or _goodCACAID2 (CAC)
   - unAIDLen == _goodGSCAID2Len (GSC) or _goodCACAID2Len (CAC)
   - ucTag == _newTag1309
   - uszValue == uszValue1309
   - punValueLen == _newDvalueLen.

   Case 1.1: If the gscBsiGcReadValue() call returns either of the codes:
   - BSI_BAD_TAG
   - BSI_IO_ERROR

then print
"gscBsiGcDataCreate() called using a too-large data value has been verified because a subsequent call to gscBsiGcReadValue() did not find the data item.
Status: Test 13.9 Passed."

Case 1.2: If the gscBsiReadValue() call returns the code BSI_OK, then print
"gscBsiGcDataCreate() called using a too-large data value has not been been verified because a subsequent call to gscBsiGcReadValue() indicated that the data value had been created.
Status: Test 13.9 Failed."

Case 1.3: If the gscBsiGcReadValue() call does not return any of the codes
- BSI_BAD_TAG
- BSI_IO_ERROR
- BSI_OK

then print
"gscBsiGcDataCreate() called using a too-large data value has not been verified because a subsequent call to gscBsiGcReadValue() was ambiguous.
Status: Test 13.9 Undetermined."

Case 2: If the gscBsiGcDataCreate() call returns the code BSI_NO_CARDSERVICE, then print
"gscBsiGcDataCreate() is not supported.
Status: Test 13.9 Not Supported."

Case 3: If the gscBsiGcDataCreate() call does not return the code BSI_NO_MORE_SPACE or the code BSI_NO_CARDSERVICE, then print
"gscBsiGcDataCreate() called using a too-large data value returned an incorrect code.
Status: Test 13.9 Failed."

Test for Assertion 13.10

The function is tested using the tag of a data item that already exists.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGcDataCreate().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1310.

3. (Pre) There exists a target container on the connected card with the following properties:
   - the ACR for the gscBsiGcDataCreate() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by AID value ==
     - _goodGSCAID1, of length _goodGSCAID1Len (GSC), or
   - _goodCACAID1, of length _goodCACAID1Len (CAC)

- there exists a data item on the target container with tag _existingTag1310 and value _existingDvalue1310
- the target container can accommodate _newDvalue1310.

4. (Pre) Print "Testing of Assertion 13.10".

5. (Pre) Establish an authenticated session with the target container on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard1310
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - strctAuthenticator == strctAuthenticator1300
   - unAuthNb == 1.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 6.

   Case 2: If the gscBsiUtilAcquireContext() call does not return the code BSI_OK, then print
   "A session cannot be established. Assertion 13.10 of gscBsiGcDataCreate() cannot be tested".
   End Test for Assertion 13.10.

6. Make a gscBsiGcDataCreate() call to the SPS, using
   - hCard == hCard1310
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - ucTag == _existingTag1310
   - uszValue == _newDvalue1310
   - unValueLen == _newDvalueLen.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_TAG_EXISTS or the return code BSI_NO_CARDSERVICE.

2. No changes are made to the container structure of the connected card.

Perform this verification by issuing a call to gscBsiGcReadValue().

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiGcDataCreate() call returns the code BSI_TAG_EXISTS, then

   Perform Test for Assertion 9.13.10.1 using
   - hCard == hCard1310.

   Verify that that the specified data value was not stored, with the specified tag, in the target container.

   Allocate the unsigned char* uszValue1310 with length _newDvalueLen.

Make a gscBsiGcReadValue() call to the SPS, using

- hCard == hCard1310
- uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
- unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
- ucTag == _existingTag1310
- uszValue == uszValue1310
- punValueLen == _newDvalueLen.

Case 1.1: If the gscBsiGcReadValue() call returns

- the code BSI_OK
- uszValue1310 == _existingDvalue1310

then print

"gscBsiGcDataCreate() called using the tag of a data item that already exists has been verified because a subsequent call to gscBsiGcReadValue() found the original data item.
Status: Test 13.10 Passed."

Case 1.2: If the gscBsiReadValue() call returns

- the code BSI_OK
- uszValue1310 /= _existingDvalue1310

then print

"gscBsiGcDataCreate() called using the tag of a data item that already exists has not been been verified because a subsequent call to gscBsiGcReadValue() indicated that the data value in question had been changed.
Status: Test 13.10 Failed."

Case 1.3: If the gscBsiGcReadValue() call does not return the code BSI_OK, then print

"gscBsiGcDataCreate() called using the tag of a data item that already exists has not been verified because a subsequent call to gscBsiGcReadValue() was ambiguous.
Status: Test 13.10 Undetermined."

Case 2: If the gscBsiGcDataCreate() call returns the code BSI_NO_CARDSERVICE, then print

"gscBsiGcDataCreate() is not supported.
Status: Test 13.10 Not Supported."

Case 2: If the gscBsiGcDataCreate() call does not return the code BSI_TAG_EXISTS or the code BSI_NO_CARDSERVICE, then print

"gscBsiGcDataCreate() called using the tag of a data item that already exists returned an incorrect code.
Status: Test 13.10 Failed."

14. gscBsiGcDataDelete()

Starting State for Each Test:

1. There exists a BSIAuthenticator array strctAuthenticator1400 with one element, the structure BSIAuthenticator1400. This structure has fields
   - unAccessMethodType1400 == BSI_AM_PIN
   - unkeyIDOrReference1400 == _keyIDOrReference1
   - uszAuthValue1400 == _goodAuthValue1
   - unAuthValueLen1400 == _goodAuthValue1Len.

Test for Assertion 14.1

The function is tested using valid parameters.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGcDataDelete().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1401.

3. (Pre) There exists a target container on the connected card with the following properties:
   - the ACR for the gscBsiGcDataDelete() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by AID value ==
     - _goodGSCAID1, of length _goodGSCAID1Len (GSC), or
   - _goodCACAID1, of length _goodCACAID1Len (CAC)
   - the target container contains at least one data item, for which
     - the tag is == _existingTag1401
     - the value is == _existingDvalue1410.

4. (Pre) Print "Testing of Assertion 14.1".

5. (Pre) Establish an authenticated session with the target container on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard1401
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - strctAuthenticator == strctAuthenticator1400
   - unAuthNb == 1.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 6.

   Case 2: If the gscBsiUtilAcquireContext() call does not return the code BSI_OK, then print
   "A session cannot be established. Assertion 14.1 of gscBsiGcDataDelete() cannot be tested".

End Test for Assertion 14.1.

6. Make a gscBsiGcDataDelete() call to the SPS, using
   - hCard == hCard1401
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - ucTag == _existingTag1401.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_OK or the return code BSI_NO_CARDSERVICE.

2. If the return code is BSI_OK, then there is no longer a data item with the tag == _existingTag1401 stored in the target container.

3. No other changes are made to the container structure of the connected card.

Note: I suppose if we were very ambitious, we would verify 3., by comparing before and after snapshots of every container on the connected card. For now, I propose limiting ourselves to verifying that the specified data value is correctly stored in the selected container.

Perform this verification by issuing a call to gscBsiGcReadValue().

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiGcDataDelete() call returns the code BSI_OK, then verify that that the specified data value has indeed been deleted from the target container.

   Allocate the unsigned char* uszValue1401 with length _existingDvalueLen.

   Make a gscBsiGcReadValue() call to the SPS, using
   - hCard == hCard1401
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - ucTag == _existingTag1401
   - uszValue == uszValue1401
   - punValueLen == _existingDvalueLen.

   Case 1.1: If the gscBsiGcReadValue() call returns
   - the code BSI_BAD_TAG

   then print
   "gscBsiGcDataDelete() called with valid parameters has been verified because a subsequent call to gscBsiGcReadValue() did not find the deleted data item.
   Status: Test 14.1 Passed."

   Case 1.2: If the gscBsiReadValue() call returns
   - the code BSI_OK

   then print
   "gscBsiGcDataDelete() called with valid parameters has been not been verified because a subsequent call to

gscBsiGcReadValue() indicated that the specified data value was not deleted.
Status: Test 14.1 Failed."

Case 1.3: If the gscBsiGcReadValue() call does not return either of the codes
- BSI_OK
- BSI_BAD_TAG

then print
"gscBsiGcDataDelete() called with valid parameters has not been verified because a subsequent call to gscBsiGcReadValue() was ambiguous.
Status: Test 14.1 Undetermined."

Case 2: If the gscBsiGcDataDelete() call returns the code BSI_NO_CARDSERVICE, then print
"gscBsiGcDataDelete() is not supported.
Status: Test 14.1 Not Supported."

Case 3: If the gscBsiGcDataDelete() call does not return the code BSI_OK or the code BSI_NO_CARDSERVICE, then print
"gscBsiGcDataDelete() called with valid parameters returned an incorrect code.
Status: Test 14.1 Failed."

Test for Assertion 14.2

The function is tested using a bad handle.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGcDataDelete().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1402.

3. (Pre) There exists a target container on the connected card with the following properties:
   - the ACR for the gscBsiGcDataDelete() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by AID value ==
     - _goodGSCAID1, of length _goodGSCAID1Len (GSC), or
   - _goodCACAID1, of length _goodCACAID1Len (CAC)
   - the target container contains at least one data item, for which
     - the tag is == _existingTag1402
     - the value is == _existingDvalue1402.

4. (Pre) Print "Testing of Assertion 14.2".

5. (Pre) Establish an authenticated session with the target container on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard1402

- uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
- unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
- strctAuthenticator == strctAuthenticator1400
- unAuthNb == 1.

Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 6.

Case 2: If the gscBsiUtilAcquireContext() call does not return the code BSI_OK, then print
"A session cannot be established. Assertion 14.2 of gscBsiGcDataDelete() cannot be tested".
End Test for Assertion 14.2.

6. Make a gscBsiGcDataDelete() call to the SPS, using
   - hCard /= hCard1402
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - ucTag == _existingTag1402.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_BAD_HANDLE or the return code BSI_NO_CARDSERVICE.

2. No changes are made to the container structure of the connected card.

Perform this verification by issuing a call to gscBsiGcReadValue().

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiGcDataDelete() call returns the code BSI_BAD_HANDLE, then:

   Perform Test for Assertion 9.14.2.1 using
   - hCard == hCard1402.

   Verify that that the specified data value was not deleted from the target container:

   Allocate the unsigned char* uszValue1402 with length _existingDvalueLen.

   Make a gscBsiGcReadValue() call to the SPS, using
   - hCard == hCard1402
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - ucTag == _existingTag1402
   - uszValue == uszValue1402
   - punValueLen == _existingDvalueLen.

   Case 1.1: If the gscBsiGcReadValue() call returns
   - the code BSI_OK
   - uszValue1402 == _existingDvalue1402

then print
"gscBsiGcDataDelete() called with a bad handle has been verified because a subsequent call to gscBsiGcReadValue() indicated that the specified data item had not been deleted.
Status: Test 14.2 Passed."

Case 1.2: If the gscBsiReadValue() call returns
- the code BSI_BAD_TAG

or
- the code BSI_OK
- uszValue1402 /= _existingDvalue1402

then print
"gscBsiGcDataDelete() called with a bad handle has been not been verified because a subsequent call to gscBsiGcReadValue() indicated that the specified data value had been deleted or otherwise changed.
Status: Test 14.2 Failed."

Case 1.3: If the gscBsiGcReadValue() call does not return either of the codes
- BSI_OK
- BSI_BAD_TAG

then print
"gscBsiGcDataDelete() called with a bad handle has not been verified because a subsequent call to gscBsiGcReadValue() was ambiguous.
Status: Test 14.2 Undetermined."

Case 2: If the gscBsiGcDataDelete() call returns the code BSI_NO_CARDSERVICE, then print
"gscBsiGcDataDelete() is not supported.
Status: Test 14.2 Not Supported."

Case 3: If the gscBsiGcDataDelete() call does not return the code BSI_BAD_HANDLE or the code BSI_NO_CARDSERVICE, then print
"gscBsiGcDataDelete() called with a bad handle returned an incorrect code.
Status: Test 14.2 Failed."

Test for Assertion 14.3

The function is tested with another application having established a transaction lock.

Until we encounter implementations that allow multiple simultaneous applications, I don't think we need to worry about this assertion.

Test for Assertion 14.4

The function is tested using a bad AID value.

Instantiation Scenario:
1. (Pre) Construct the Starting State for the testing of gscBsiGcDataDelete().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1404.

3. (Pre) There exists a target container on the connected card with the following properties:
   - the ACR for the gscBsiGcDataDelete() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by AID value ==
     - _goodGSCAID1, of length _goodGSCAID1Len (GSC), or
   - _goodCACAID1, of length _goodCACAID1Len (CAC)
   - the target container contains at least one data item, for which
     - the tag is == _existingTag1404
     - the value is == _existingDvalue1404.

4. (Pre) There does not exist a container on the card with AID value == _badGSCAID (GSC) or _badCACAID (CAC).

5. (Pre) Print "Testing of Assertion 14.4".

6. (Pre) Establish an authenticated session with the target container on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard1404
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - strctAuthenticator == strctAuthenticator1400
   - unAuthNb == 1.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 6.

   Case 2: If the gscBsiUtilAcquireContext() call does not return the code BSI_OK, then print
   "A session cannot be established. Assertion 14.4 of gscBsiGcDataDelete() cannot be tested".
   End Test for Assertion 14.4.

7. Make a gscBsiGcDataDelete() call to the SPS, using
   - hCard == hCard1404
   - uszAID == _badGSCAID (GSC) or _badCACAID (CAC)
   - unAIDLen == _badGSCAIDLen (GSC) or _badCACAIDLen (CAC)
   - ucTag == _existingTag1404.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_BAD_AID or the return code BSI_NO_CARDSERVICE.

2. No changes are made to the container structure of the connected card.

Perform this verification by issuing a call to gscBsiGcReadValue().

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiGcDataDelete() call returns the code BSI_BAD_AID, then:

   Perform Test for Assertion 9.14.4.1 using
   - hCard == hCard1404.

   Verify that that the specified data value was not deleted from the target container:

   Allocate the unsigned char* uszValue1404 with length _existingDvalue1404.

   Make a gscBsiGcReadValue() call to the SPS, using
   - hCard == hCard1404
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - ucTag == _existingTag1404
   - uszValue == uszValue1404
   - punValueLen == _existingDvalueLen.

   Case 1.1: If the gscBsiGcReadValue() call returns
   - the code BSI_OK
   - uszValue1404 == _existingDvalue1404

   then print
   "gscBsiGcDataDelete() called with a bad AID value has been verified because a subsequent call to gscBsiGcReadValue() indicated that the specified data item had not been deleted.
   Status: Test 14.4 Passed."

   Case 1.2: If the gscBsiReadValue() call returns
   - the code BSI_BAD_TAG

   or
   - the code BSI_OK
   - uszValue1404 /= _existingDvalue1404

   then print
   "gscBsiGcDataDelete() called with a bad AID value has been not been verified because a subsequent call to gscBsiGcReadValue() indicated that the specified data value had been deleted or otherwise changed.
   Status: Test 14.4 Failed."

   Case 1.3: If the gscBsiGcReadValue() call does not return either of the codes
   - BSI_OK
   - BSI_BAD_TAG

   then print
   "gscBsiGcDataDelete() called with a bad AID value has not been verified because a subsequent call to gscBsiGcReadValue() was ambiguous.
   Status: Test 14.4 Undetermined."

Case 2: If the gscBsiGcDataDelete() call returns the code BSI_NO_CARDSERVICE, then print
"gscBsiGcDataDelete() is not supported.
Status: Test 14.4 Not Supported."

Case 3: If the gscBsiGcDataDelete() call does not return the code BSI_BAD_AID or the code BSI_NO_CARDSERVICE, then print
"gscBsiGcDataDelete() called with a bad AID value returned an incorrect code.
Status: Test 14.4 Failed."

Test for Assertion 14.5

The function is tested using a bad AID length.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGcDataDelete().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1405.

3. (Pre) There exists a target container on the connected card with the following properties:
   - the ACR for the gscBsiGcDataDelete() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by AID value ==
     - _goodGSCAID1, of length _goodGSCAID1Len (GSC), or
   - _goodCACAID1, of length _goodCACAID1Len (CAC)
   - the target container contains at least one data item, for which
     - the tag is == _existingTag1405
     - the value is == _existingDvalue1405.

4. (Pre) Print "Testing of Assertion 14.5".

5. (Pre) Establish an authenticated session with the target container on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard1405
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - strctAuthenticator == strctAuthenticator1400
   - unAuthNb == 1.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 6.

   Case 2: If the gscBsiUtilAcquireContext() call does not return the code BSI_OK, then print
   "A session cannot be established. Assertion 14.5 of gscBsiGcDataDelete() cannot be tested".
   End Test for Assertion 14.5.

6. Make a gscBsiGcDataDelete() call to the SPS, using
   - hCard == hCard1405
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _badAIDLen
   - ucTag == _existingTag1405.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_BAD_PARAM or the return code BSI_NO_CARDSERVICE.

2. No changes are made to the container structure of the connected card.

Perform this verification by issuing a call to gscBsiGcReadValue().

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiGcDataDelete() call returns the code BSI_BAD_PARAM, then:

   Perform Test for Assertion 9.14.5.1 using
   - hCard == hCard1405.

   Verify that that the specified data value was not deleted from the target container:

   Allocate the unsigned char* uszValue1405 with length _existingDvalueLen.

   Make a gscBsiGcReadValue() call to the SPS, using
   - hCard == hCard1405
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - ucTag == _existingTag1405
   - uszValue == uszValue1405
   - punValueLen == _existingDvalueLen.

   Case 1.1: If the gscBsiGcReadValue() call returns
   - the code BSI_OK
   - uszValue1405 == _existingDvalue1405

   then print
   "gscBsiGcDataDelete() called with a bad AID length has been verified because a subsequent call to gscBsiGcReadValue() indicated that the specified data item had not been deleted.
   Status: Test 14.5 Passed."

   Case 1.2: If the gscBsiReadValue() call returns
   - the code BSI_BAD_TAG

   or
   - the code BSI_OK
   - uszValue1405 /= _existingDvalue1405

   then print

"gscBsiGcDataDelete() called with a bad AID length has been not been verified because a subsequent call to gscBsiGcReadValue() indicated that the specified data value had been deleted or otherwise changed.
Status: Test 14.5 Failed."

Case 1.3: If the gscBsiGcReadValue() call does not return either of the codes
- BSI_OK
- BSI_BAD_TAG

then print
"gscBsiGcDataDelete() called with a bad AID length has not been verified because a subsequent call to gscBsiGcReadValue() was ambiguous.
Status: Test 14.5 Undetermined."

Case 2: If the gscBsiGcDataDelete() call returns the code BSI_NO_CARDSERVICE, then print
"gscBsiGcDataDelete() is not supported.
Status: Test 14.5 Not Supported."

Case 3: If the gscBsiGcDataDelete() call does not return the code BSI_BAD_PARAM or the code BSI_NO_CARDSERVICE, then print
"gscBsiGcDataDelete() called with a bad AID length returned an incorrect code.
Status: Test 14.5 Failed."

Test for Assertion 14.6

The function is tested using a bad tag.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGcDataDelete().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1406.

3. (Pre) There exists a target container on the connected card with the following properties:
   - the ACR for the gscBsiGcDataDelete() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by AID value ==
     - _goodGSCAID1, of length _goodGSCAID1Len (GSC), or
   - _goodCACAID1, of length _goodCACAID1Len (CAC)
   - the target container does not contain a data item for which the tag is == _newTag1406.

4. (Pre) Print "Testing of Assertion 14.6".

5. (Pre) Establish an authenticated session with the target container on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard1406

- uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
- unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
- strctAuthenticator == strctAuthenticator1400
- unAuthNb == 1.

Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 6.

Case 2: If the gscBsiUtilAcquireContext() call does not return the code BSI_OK, then print
"A session cannot be established. Assertion 14.6 of gscBsiGcDataDelete() cannot be tested".
End Test for Assertion 14.6.

6. Make a gscBsiGcDataDelete() call to the SPS, using
   - hCard == hCard1406
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - ucTag == _newTag1406.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_BAD_TAG or the return code BSI_NO_CARDSERVICE.

2. No changes are made to the container structure of the connected card.

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiGcDataDelete() call returns the code BSI_BAD_TAG, then:

   Perform Test for Assertion 9.14.6.1 using
   - hCard == hCard1406.

   Print
   "gscBsiGcDataDelete() called with a bad tag has been verified.
   Status: Test 14.6 Passed."

   Case 2: If the gscBsiGcDataDelete() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiGcDataDelete() is not supported.
   Status: Test 14.6 Not Supported."

   Case 3: If the gscBsiGcDataDelete() call does not return the code BSI_BAD_TAG or the code BSI_NO_CARDSERVICE, then print
   "gscBsiGcDataDelete() called with a bad tag returned an incorrect code.
   Status: Test 14.6 Failed."

Test for Assertion 14.7

The function is tested using a card that has been removed.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGcDataDelete().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1407.

3. (Pre) There exists a target container on the connected card with the following properties:
   - the ACR for the gscBsiGcDataDelete() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by AID value ==
     - _goodGSCAID1, of length _goodGSCAID1Len (GSC), or
   - _goodCACAID1, of length _goodCACAID1Len (CAC)
   - the target container contains at least one data item, for which
     - the tag is == _existingTag1407
     - the value is == _existingDvalue1407.

4. (Pre) Print "Testing of Assertion 14.7".

5. (Pre) Establish an authenticated session with the target container on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard1407
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - strctAuthenticator == strctAuthenticator1400
   - unAuthNb == 1.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 6.

   Case 2: If the gscBsiUtilAcquireContext() call does not return the code BSI_OK, then print
   "A session cannot be established. Assertion 14.7 of gscBsiGcDataDelete() cannot be tested".
   End Test for Assertion 14.7.

6. (Pre) Remove the connected card from the reader.

7. Make a gscBsiGcDataDelete() call to the SPS, using
   - hCard == hCard1407
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - ucTag == _existingTag1407.

Verification Goal:

To verify the Expected Results:

1. The call returns
   - the return code BSI_CARD_REMOVED or the return code BSI_NO_CARDSERVICE.

2. No changes are made to the container structure of the connected card.

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiGcDataDelete() call returns the code BSI_CARD_REMOVED, then print
   "gscBsiGcDataDelete() called with the connected card removed has been verified.
   Status: Test 14.7 Passed."

   Case 2: If the gscBsiGcDataDelete() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiGcDataDelete() is not supported.
   Status: Test 14.7 Not Supported."

   Case 3: If the gscBsiGcDataDelete() call does not return the code BSI_CARD_REMOVED or the code BSI_NO_CARDSERVICE, then print
   "gscBsiGcDataDelete() called with the connected card removed returned an incorrect code.
   Status: Test 14.7 Failed."

Test for Assertion 14.8

The function is tested without fulfilling the applicable ACR.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGcDataDelete().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1408.

3. (Pre) There exists a target container on the connected card with the following properties:
   - the ACR for the gscBsiGcDataDelete() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by AID value ==
     - _goodGSCAID1, of length _goodGSCAID1Len (GSC), or
     - _goodCACAID1, of length _goodCACAID1Len (CAC)
   - this container contains at least one data item, for which
     - the tag is == _existingTag1408
     - the value is == _existingDvalue1408.

4. (Pre) Print "Testing of Assertion 14.8".

5. (Pre) Ensure that there is no authenticated session with the target container:

   Make a gscBsiUtilReleaseContext() call to the SPS, using
   - hCard == hCard1408
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC).

6. Make a gscBsiGcDataDelete() call to the SPS, using

- hCard == hCard1408
- uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
- unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
- ucTag == _existingTag1408.

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_ACCESS_DENIED or the return code BSI_NO_CARDSERVICE.

2. No changes are made to the container structure of the connected card.

Perform this verification by issuing a call to gscBsiGcReadValue().

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiGcDataDelete() call returns the code BSI_ACCESS_DENIED, then:

   Perform Test for Assertion 9.14.8.1 using
   - hCard == hCard1408.

   Verify that that the specified data value was not deleted from the second container:

   Establish an authenticated session with the target container on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard1408
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - strctAuthenticator == strctAuthenticator1400
   - unAuthNb == 1.

   Case 1.1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then

   Allocate the unsigned char* uszValue1408 with length _existingDvalueLen.

   Make a gscBsiGcReadValue() call to the SPS, using
   - hCard == hCard1408
   - uszAID == _goodGSCAID2 (GSC) or _goodCACAID2 (CAC)
   - unAIDLen == _goodGSCAID2Len (GSC) or _goodCACAID2Len (CAC)
   - ucTag == _existingTag1408
   - uszValue == uszValue1408
   - punValueLen == _existingDvalueLen.

   Case 1.1.1: If the gscBsiGcReadValue() call returns
   - the code BSI_OK
   - uszValue1408 == _existingDvalue1408

   then print

"gscBsiGcDataDelete() called without fulfilling the applicable ACR has been verified because a subsequent call to gscBsiGcReadValue() indicated that the specified data item had not been deleted.
Status: Test 14.8 Passed."

Case 1.1.2: If the gscBsiReadValue() call returns
- the code BSI_BAD_TAG

or
- the code BSI_OK
- uszValue1408 /= _existingDvalue1408

then print
"gscBsiGcDataDelete() called without fulfilling the applicable ACR has been not been verified because a subsequent call to gscBsiGcReadValue() indicated that the specified data value had been deleted or otherwise changed.
Status: Test 14.8 Failed."

Case 1.1.3: If the gscBsiGcReadValue() call does not return either of the codes
- BSI_OK
- BSI_BAD_TAG

then print
"gscBsiGcDataDelete() called without fulfilling the applicable ACR has not been verified because a subsequent call to gscBsiGcReadValue() was ambiguous.
Status: Test 14.8 Undetermined."

Case 1.2: If the gscBsiUtilAcquireContext call does not return the code BSI_OK, then print
"A session could not be established.
gscBsiUtilDataDelete cannot be verified.
Status: Test 14.8 Undetermined."

Case 2: If the gscBsiGcDataDelete() call returns the code BSI_NO_CARDSERVICE, then print
"gscBsiGcDataDelete() is not supported.
Status: Test 14.8 Not Supported."

Case 3: If the gscBsiGcDataDelete() call does not return the code BSI_ACCESS_DENIED or the code BSI_NO_CARDSERVICE, then print
"gscBsiGcDataDelete() called without fulfilling the applicable ACR returned an incorrect code.
Status: Test 14.8 Failed."

15. gscBsiGcGetContainerProperties()

Starting State for Each Test:

1. There exists a GCAcr variable strctGCacr1500, with members
   - BSIAcr strctCreateACR1500, with members
     - unsigned long unCreateACRType1500
     - unsigned long unCreateKeyIDOrReference1500
     - unsigned long unCreateAuthID1500
     - unsigned long unCreateACRID1500
   - BSIAcr strctDeleteACR1500
     - unsigned long unDeleteACRType1500
     - unsigned long unDeleteKeyIDOrReference1500
     - unsigned long unDeleteAuthID1500
     - unsigned long unDeleteACRID1500
   - BSIAcr strctReadTagListACR1500
     - unsigned long unReadTagListACRType1500
     - unsigned long unReadTagListKeyIDOrReference1500
     - unsigned long unReadTagListAuthID1500
     - unsigned long unReadTagListACRID1500
   - BSIAcr strctReadValueACR1500
     - unsigned long unReadValueACRType1500
     - unsigned long unReadValueKeyIDOrReference1500
     - unsigned long unReadValueAuthID1500
     - unsigned long unReadValueACRID1500
   - BSIAcr strctUpdateValueACR1500
     - unsigned long unUpdateValueACRType1500
     - unsigned long unUpdateValueKeyIDOrReference1500
     - unsigned long unUpdateValueAuthID1500
     - unsigned long unUpdateValueACRID1500.

2. There exists a GCContainerSize variable strctContainerSizes1500, with members
   - unMaxNbDataItems1500
   - unMaxValueStorageSize1500.

3. There exists a string containerVersion1500.

Test for Assertion 15.1

The function is tested using valid parameters.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGcGetContainerProperties().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1501.

3. (Pre) There exists a target container on the connected card with the following properties:

- the ACR for each of the gscBsiGcDataCreate(), gscBsiGcDataDelete(), gscBsiGcReadTagList(), gscBsiGcReadValue(), and gscBsiGcDataUpdate() services has
  - access method type == BSI_ACR_PIN
  - key ID or reference == _keyIDOrReference1
  - number of access methods logically combined in the ACR == 1
  - ACRID == null
- this container is represented by AID value ==
  - _goodGSCAID1, of length _goodGSCAID1Len (GSC), or
  - _goodCACAID1, of length _goodCACAID1Len (CAC).

4. (Pre) Print "Testing of Assertion 15.1".

5. Make a gscBsiGcGetContainerPropereties() call to the SPS, using
   - hCard == hCard1501
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - strctGCacr == strctGCacr1500
   - strctContainerSizes == strctContainerSizes1500
   - containerVersion == containerVersion1500.

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_OK or the return code BSI_NO_CARDSERVICE.

2. If the return code is BSI_OK, then the variables of strctGCacr1500 are correctly set to indicate access control conditions for all operations.

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiGcGetContainerProperties() call returns the code BSI_OK, then:

   Case 1.1: If
   - each of unCreateACRType1500, unDeleteACRType1500, unReadTagListACRType1500, unReadValueACRType1500, and unUpdateValueACRType1500 == BSI_ACR_PIN

   and
   - each of unCreateKeyIDOrReference1500, unDeleteKeyIDOrReference1500, unReadTagListKeyIDOrReference1500, unReadValueKeyIDOrReference1500, and unUpdateKeyIDOrReference1500 == _keyIDOrReference1

   and
   - each of unCreateAuthNB1500, unDeleteAuthNB1500, unReadTagListAuthNB1500, unReadValueAuthNB1500, and unUpdateValueAuthNB1500 == 1

   and
   - each of unCreateACRID1500, unDeleteACRID1500, unReadTagListACRID1500, unReadValueACRID1500, and unUpdateValueACRID1500 == null

   then print
   "gscBsiGcGetContainerProperties() called with valid parameters has been verified.

Status: Test 15.1 Passed."

Case 1.2: If
- any of unCreateACRType1500, unDeleteACRType1500, unReadTagListACRType1500, unReadValueACRType1500, and unUpdateValueACRType1500 /= BSI_ACR_PIN

or
- any of unCreateKeyIDOrReference1500, unDeleteKeyIDOrReference1500, unReadTagListKeyIDOrReference1500, unReadValueKeyIDOrReference1500, and unUpdateKeyIDOrReference1500 /= _keyIDOrReference1

or
- any of unCreateAuthNB1500, unDeleteAuthNB1500, unReadTagListAuthNB1500, unReadValueAuthNB1500, and unUpdateValueAuthNB1500 /= 1

or
- any of unCreateACRID1500, unDeleteACRID1500, unReadTagListACRID1500, unReadValueACRID1500, and unUpdateValueACRID1500 /= null

then print
"gscBsiGcGetContainerProperties() called with valid parameters has not been verified.
Status: Test 15.1 Failed."

Case 2: If the gscBsiGcGetContainerProperties() call returns the code BSI_NO_CARDSERVICE, then print
"gscBsiGcGetContainerProperties() is not supported.
Status: Test 15.1 Not Supported."

Case 3: If the gscBsiGcGetContainerProperties() call does not return the code BSI_OK or the code BSI_NO_CARDSERVICE, then print
"gscBsiGcGetContainerProperties() called with valid parameters returned an incorrect code.
Status: Test 15.1 Failed."

Test for Assertion 15.2

The function is tested using a bad handle.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGcGetContainerProperties().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1502.

3. (Pre) There exists a target container on the connected card with the following properties:
   - the ACR for each of the gscBsiGcDataCreate(), gscBsiGcDataDelete(), gscBsiGcReadTagList(), gscBsiGcReadValue(), and gscBsiGcDataUpdate() services has
     - access method type == BSI_ACR_PIN
     - key ID or reference == _keyIDOrReference1

- number of access methods logically combined in the ACR == 1
- ACRID == null
- this container is represented by AID value ==
  - _goodGSCAID1, of length _goodGSCAID1Len (GSC), or
  - _goodCACAID1, of length _goodCACAID1Len (CAC).

4. (Pre) Print "Testing of Assertion 15.2".

5. Make a gscBsiGcGetContainerProperties() call to the SPS, using
   - hCard /= hCard1502
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - strctGCacr == strctGCacr1500
   - strctContainerSizes == strctContainerSizes1500
   - containerVersion == containerVersion1500.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_BAD_HANDLE or the return code BSI_NO_CARDSERVICE.

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiGcGetContainerProperties() call returns the code BSI_BAD_HANDLE, then:

   Perform Test for Assertion 9.15.2.1 using
   - hCard == hCard1502.

   Print
   "gscBsiGcGetContainerProperties() called with a bad handle has been verified.
   Status: Test 15.2 Passed."

   Case 2: If the gscBsiGcGetContainerProperties() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiGcGetContainerProperties() is not supported.
   Status: Test 15.2 Not Supported."

   Case 3: If the gscBsiGcGetContainerProperties() call does not return the code BSI_BAD_HANDLE or the code BSI_NO_CARDSERVICE, then print
   "gscBsiGcGetContainerProperties() called with a bad handle returned an incorrect code.
   Status: Test 15.2 Failed."

Test for Assertion 15.3

The function is tested with another application having established a transaction lock.

Until we encounter implementations that allow multiple simultaneous applications, I don't think we need to worry about this assertion.

Test for Assertion 15.4

The function is tested using a bad AID value.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGcGetContainerProperties().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1504.

3. (Pre) There exists a target container on the connected card with the following properties:
   - the ACR for each of the gscBsiGcDataCreate(), gscBsiGcDataDelete(), gscBsiGcReadTagList(), gscBsiGcReadValue(), and gscBsiGcDataUpdate() services has
     - access method type == BSI_ACR_PIN
     - key ID or reference == _keyIDOrReference1
     - number of access methods logically combined in the ACR == 1
     - ACRID == null
   - this container is represented by AID value ==
     - _goodGSCAID1, of length _goodGSCAID1Len (GSC), or
     - _goodCACAID1, of length _goodCACAID1Len (CAC).

4. (Pre) There does not exist a container on the card with AID value == _badGSCAID (GSC) or _badCACAID (CAC).

5. (Pre) Print "Testing of Assertion 15.4".

6. Make a gscBsiGcGetContainerProperties() call to the SPS, using
   - hCard == hCard1504
   - uszAID == _badGSCAID (GSC) or _badCACAID (CAC)
   - unAIDLen == _badGSCAIDLen (GSC) or _badCACAIDLen (CAC)
   - strctGCacr == strctGCacr1500
   - strctContainerSizes == strctContainerSizes1500
   - containerVersion == containerVersion1500.

Verification Goal:

To verify the Expected Results:

1. The call returns
   - the return code BSI_BAD_AID or the return code BSI_NO_CARDSERVICE.

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiGcGetContainerProperties() call returns the code BSI_BAD_AID, then:

   Perform Test for Assertion 9.15.4.1 using
   - hCard == hCard1504.

   Print
   "gscBsiGcGetContainerProperties() called with a bad AID value has been verified.
   Status: Test 15.4 Passed."

Case 2: If the gscBsiGcGetContainerProperties() call returns the code BSI_NO_CARDSERVICE, then print
"gscBsiGcGetContainerProperties() is not supported.
Status: Test 15.4 Not Supported."

Case 3: If the gscBsiGcGetContainerProperties() call does not return the code BSI_BAD_AID or the code BSI_NO_CARDSERVICE, then print
"gscBsiGcGetContainerProperties() called with a bad AID value returned an incorrect code.
Status: Test 15.4 Failed."

Test for Assertion 15.5

The function is tested using a bad AID length.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGcGetContainerProperties().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1505.

3. (Pre) There exists a target container on the connected card with the following properties:
   - the ACR for each of the gscBsiGcDataCreate(), gscBsiGcDataDelete(), gscBsiGcReadTagList(), gscBsiGcReadValue(), and gscBsiGcDataUpdate() services has
     - access method type == BSI_ACR_PIN
     - key ID or reference == _keyIDOrReference1
     - number of access methods logically combined in the ACR == 1
     - ACRID == null
   - this container is represented by AID value ==
     - _goodGSCAID1, of length _goodGSCAID1Len (GSC), or
     - _goodCACAID1, of length _goodCACAID1Len (CAC).

4. (Pre) Print "Testing of Assertion 15.5".

5. Make a gscBsiGcGetContainerProperties() call to the SPS, using
   - hCard == hCard1505
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == __badAIDLen
   - strctGCacr == strctGCacr1500
   - strctContainerSizes == strctContainerSizes1500
   - containerVersion == containerVersion1500.

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_BAD_PARAM or the return code BSI_NO_CARDSERVICE.

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiGcGetContainerProperties() call returns the code BSI_BAD_PARAM, then:

   Perform Test for Assertion 9.15.5.1 using
   - hCard == hCard1505.

   Print
   "gscBsiGcGetContainerProperties() called with a bad AID length has been verified.
   Status: Test 15.5 Passed."

   Case 2: If the gscBsiGcGetContainerProperties() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiGcGetContainerProperties() is not supported.
   Status: Test 15.5 Not Supported."

   Case 3: If the gscBsiGcGetContainerProperties() call does not return the code BSI_BAD_PARAM or the code BSI_NO_CARDSERVICE, then print
   "gscBsiGcGetContainerProperties() called with a bad AID length returned an incorrect code.
   Status: Test 15.5 Failed."

Test for Assertion 15.6

The function is tested with a removed card.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGcGetContainerProperties().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1506.

3. (Pre) There exists a target container on the connected card with the following properties:
   - the ACR for each of the gscBsiGcDataCreate(), gscBsiGcDataDelete(), gscBsiGcReadTagList(), gscBsiGcReadValue(), and gscBsiGcDataUpdate() services has
     - access method type == BSI_ACR_PIN
     - key ID or reference == _keyIDOrReference1
     - number of access methods logically combined in the ACR == 1
     - ACRID == null
   - this container is represented by AID value ==
     - _goodGSCAID1, of length _goodGSCAID1Len (GSC), or
     - _goodCACAID1, of length _goodCACAID1Len (CAC).

4. (Pre) Print "Testing of Assertion 15.6".

5. Remove the connected card from the reader.

6. Make a gscBsiGcGetContainerProperties() call to the SPS, using
   - hCard == hCard1506
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)

- unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
- strctGCacr == strctGCacr1500
- strctContainerSizes == strctContainerSizes1500
- containerVersion == containerVersion1500.

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_CARD_REMOVED or the return code BSI_NO_CARDSERVICE.

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiGcGetContainerProperties() call returns the code BSI_CARD_REMOVED, then print
   "gscBsiGcGetContainerProperties() called with the connected card removed has been verified.
   Status: Test 15.6 Passed."

   Case 2: If the gscBsiGcGetContainerProperties() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiGcGetContainerProperties() is not supported.
   Status: Test 15.6 Not Supported."

   Case 3: If the gscBsiGcGetContainerProperties() call does not return the code BSI_CARD_REMOVED or the code BSI_NO_CARDSERVICE, then print
   "gscBsiGcGetContainerProperties() called with the connected card removed returned an incorrect code.
   Status: Test 15.6 Failed."

16. gscBsiGcReadTagList()

Starting State for Each Test:

1. There exists a BSIAuthenticator array strctAuthenticator1600 with one element, the structure BSIAuthenticator1600. This structure has fields
   - unAccessMethodType1600 == BSI_AM_PIN
   - unkeyIDOrReference1600 == _keyIDOrReference1
   - uszAuthValue1600 == _goodAuthValue1
   - unAuthValueLen1600 == _goodAuthValue1Len.

Test for Assertion 16.1

The function is tested using valid parameters (Discovery Method 1).

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGcReadTagList().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1601.

3. (Pre) There exists a target container on the connected card with the following properties:
   - the ACR for the gscBsiGcReadTagList() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by AID value ==
     - _goodGSCAID1, of length _goodGSCAID1Len (GSC), or
     - _goodCACAID1, of length _goodCACAID1Len (CAC)
   - the container contains a collection of 27 data items whose tags are:
     - _existingTag1309
     - _existingTag1310
     - _existingTag1401
     - _existingTag1402
     - _existingTag1403
     - _existingTag1404
     - _existingTag1405
     - _existingTag1406
     - _existingTag1407
     - _existingTag1408
     - _existingTag1701
     - _existingTag1702
     - _existingTag1703
     - _existingTag1704
     - _existingTag1706
     - _existingTag1707
     - _existingTag1709
     - _existingTag1710
     - _existingTag1801
     - _existingTag1802
     - _existingTag1803
     - _existingTag1804
     - _existingTag1805

- _existingTag1806
- _existingTag1807
- _existingTag1808
- _existingTag1809.

4. (Pre) Print "Testing of Assertion 16.1".

5. (Pre) Establish an authenticated session with the target container on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard1601
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - strctAuthenticator == strctAuthenticator1600
   - unAuthNb == 1.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 6.

   Case 2: If the gscBsiUtilAcquireContext() call does not return the code BSI_OK, then print
   "A session cannot be established. Assertion 16.1 of gscBsiGcDataCreate() cannot be tested".
   End Test for Assertion 16.1.

6. (Pre) Allocate the unsigned long* punNbTags1601 == 0.

7. Make a gscBsiGcReadTagList() call to the SPS, using
   - hCard == hCard1601
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - tagArray == NULL
   - punNbTags == punNbTags1601.

   Case 1: If the gscBsiGcReadTagList() call returns the code BSI_OK, then print
   "gscBsiGcReadTagList() (Discovery Method 1, Discovery Mode) succeeded."
   Continue with 8.

   Case 2: If the gscBsiGcReadTagList() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiGcReadTagList() is not supported.
   Status: Test 16.1 Not Supported."
   End Test for Assertion 16.1.

   Case 3: If the gscBsiGcReadTagList() call does not return the code BSI_OK or the code BSI_NO_CARDSERVICE, then print
   "gscBsiGcReadTagList() (Discovery Method 1, Discovery Mode) returned an invalid code.
   Status: Test 16.1 Failed."
   End Test for Assertion 16.1.

8. Allocate the unsigned GCtag array tagArray1601 with length == punNBTags1601*.

9. Make a gscBsiGcReadTagList() call to the SPS, using
   - hCard == hCard1601
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - tagArray == tagArray1601
   - punNbTags == punNbTags1601.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_OK or the return code BSI_NO_CARDSERVICE.

2. If the return code is BSI_OK, then tagArray1601 == an array containing the list of tags for the target container.

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiGcReadTagList() call returns the code BSI_OK, then:

   Case 1.1: If tagArray1601 contains precisely the 27 elements:
   - o _existingTag1309
   - o _existingTag1310
   - o _existingTag1401
   - o _existingTag1402
   - o _existingTag1403
   - o _existingTag1404
   - o _existingTag1405
   - o _existingTag1406
   - o _existingTag1407
   - o _existingTag1408
   - o _existingTag1701
   - o _existingTag1702
   - o _existingTag1703
   - o _existingTag1704
   - o _existingTag1706
   - o _existingTag1707
   - o _existingTag1709
   - o _existingTag1710
   - o _existingTag1801
   - o _existingTag1802
   - o _existingTag1803
   - o _existingTag1804
   - o _existingTag1805
   - o _existingTag1806
   - o _existingTag1807
   - o _existingTag1808
   - o _existingTag1809

   then print
   "gscBsiGcReadTagList() with a called with valid parameters has been verified.
   Status: Test 16.1 Passed."

   Case 1.2: If tagArray1601 does not contain precisely the 27 elements:
   - o _existingTag1309
   - o _existingTag1310

- o _existingTag1401
- o _existingTag1402
- o _existingTag1403
- o _existingTag1404
- o _existingTag1405
- o _existingTag1406
- o _existingTag1407
- o _existingTag1408
- o _existingTag1701
- o _existingTag1702
- o _existingTag1703
- o _existingTag1704
- o _existingTag1706
- o _existingTag1707
- o _existingTag1709
- o _existingTag1710
- o _existingTag1801
- o _existingTag1802
- o _existingTag1803
- o _existingTag1804
- o _existingTag1805
- o _existingTag1806
- o _existingTag1807
- o _existingTag1808
- o _existingTag1809

then print
"gscBsiGcReadTagList() (Discovery Method 1, Final Mode) called with valid parameters has not been verified.
Status: Test 16.1 Failed."

Case 2: If the gscBsiGcReadTagList() call returns the code BSI_NO_CARDSERVICE, then print
"gscBsiGcReadTagList() is not supported.
Status: Test 16.2 Not Supported."

Case 3: If the gscBsiGcReadTagList() call does not return the code BSI_OK or the code BSI_NO_CARDSERVICE, then print
"gscBsiGcReadTagList() (Discovery Method 1, Final Mode) called with valid parameters returned an incorrect code.
Status: Test 16.1 Failed."

Test for Assertion 16.2

The function is tested using valid parameters (Discovery Method 2).

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGcReadTagList().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1602.

3. (Pre) There exists a target container on the connected card with the following properties:
   - the ACR for the gscBsiGcReadTagList() service has the value BSI_ACR_PIN

- the value of the PIN is _PIN
- the container is represented by AID value ==
  - _goodGSCAID1, of length _goodGSCAID1Len (GSC), or
- _goodCACAID1, of length _goodCACAID1Len (CAC)
- the container contains a collection of 27 data items whose tags are:
  - _existingTag1309
  - _existingTag1310
  - _existingTag1401
  - _existingTag1402
  - _existingTag1403
  - _existingTag1404
  - _existingTag1405
  - _existingTag1406
  - _existingTag1407
  - _existingTag1408
  - _existingTag1701
  - _existingTag1702
  - _existingTag1703
  - _existingTag1704
  - _existingTag1706
  - _existingTag1707
  - _existingTag1709
  - _existingTag1710
  - _existingTag1801
  - _existingTag1802
  - _existingTag1803
  - _existingTag1804
  - _existingTag1805
  - _existingTag1806
  - _existingTag1807
  - _existingTag1808
  - _existingTag1809.

4. (Pre) Print "Testing of Assertion 16.2".

5. (Pre) Establish an authenticated session with the target container on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard1602
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - strctAuthenticator == strctAuthenticator1600
   - unAuthNb == 1.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 6.

   Case 2: If the gscBsiUtilAcquireContext() call does not return the code BSI_OK, then print
   "A session cannot be established. Assertion 16.2 of gscBsiGcDataCreate() cannot be tested".
   End Test for Assertion 16.2.

6. (Pre) Allocate the unsigned GCtag array tagArray1602 with length == 0.

7. (Pre) Allocate the unsigned long* punNbTags1602 == 0.

8. Make a gscBsiGcReadTagList() call to the SPS, using
   - hCard == hCard1602
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - tagArray == tagArray1602
   - punNbTags == punNbTags1602.

   Case 1: If the gscBsiGcReadTagList() call returns the code BSI_INSUFFICIENT_BUFFER, then print
   "gscBsiGcReadTagList() (Discovery Method 2, Discovery Mode) correctly returned the code BSI_INSUFFICIENT_BUFFER."
   Continue with 9.

   Case 2: If the gscBsiGcReadTagList() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiGcReadTagList() is not supported.
   Status: Test 16.2 Not Supported."
   End Test for Assertion 16.2.

   Case 3: If the gscBsiGcReadTagList() call does not return the code BSI_INSUFFICIENT_BUFFER or the code BSI_NO_CARDSERVICE, then print
   "gscBsiGcReadTagList() (Discovery Method 2, Discovery Mode) returned an incorrect code.
   Status: Test 16.2 Failed.
   End Test for Assertion 16.2.

9. Re-allocate the unsigned GCtag array tagArray1602 with length == punNbTags1602.

10. Make a gscBsiGcReadTagList() call to the SPS, using
    - hCard == hCard1602
    - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
    - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
    - tagArray == tagArray1602
    - punNbTags == punNbTags1602.

<u>Verification Goal</u>:
To verify the Expected Results:
1. The call returns
   - the return code BSI_OK or the return code BSI_NO_CARDSERVICE.

2. If the return code is BSI_OK, then tagArray1602 == an array containing the list of tags for the target container.

<u>Verification and Reporting Scenario</u>:
2. **Case 1**: If the gscBsiGcReadTagList() call returns the code BSI_OK, then:

   Case 1.1: If tagArray1602 contains precisely the 27 elements:
   - o _existingTag1309
   - o _existingTag1310

- o _existingTag1401
- o _existingTag1402
- o _existingTag1403
- o _existingTag1404
- o _existingTag1405
- o _existingTag1406
- o _existingTag1407
- o _existingTag1408
- o _existingTag1701
- o _existingTag1702
- o _existingTag1703
- o _existingTag1704
- o _existingTag1706
- o _existingTag1707
- o _existingTag1709
- o _existingTag1710
- o _existingTag1801
- o _existingTag1802
- o _existingTag1803
- o _existingTag1804
- o _existingTag1805
- o _existingTag1806
- o _existingTag1807
- o _existingTag1808
- o _existingTag1809

then print
"gscBsiGcReadTagList() with a called with valid parameters has been verified.
Status: Test 16.2 Passed."

Case 1.2: If tagArray1602 does not contain precisely the 27 elements:

- o _existingTag1309
- o _existingTag1310
- o _existingTag1401
- o _existingTag1402
- o _existingTag1403
- o _existingTag1404
- o _existingTag1405
- o _existingTag1406
- o _existingTag1407
- o _existingTag1408
- o _existingTag1701
- o _existingTag1702
- o _existingTag1703
- o _existingTag1704
- o _existingTag1706
- o _existingTag1707
- o _existingTag1709
- o _existingTag1710
- o _existingTag1801
- o _existingTag1802
- o _existingTag1803
- o _existingTag1804
- o _existingTag1805
- o _existingTag1806
- o _existingTag1807

- o _existingTag1808
- o _existingTag1809

Case 2: If the gscBsiGcReadTagList() call returns the code BSI_NO_CARDSERVICE, then print
"gscBsiGcReadTagList() is not supported.
Status: Test 16.2 Not Supported."

Case 3: If the gscBsiGcReadTagList() call does not return the code BSI_OK or the code BSI_NO_CARDSERVICE, then print
"gscBsiGcReadTagList() (Discovery Method 2, Final Mode) called with valid parameters returned an incorrect code.
Status: Test 16.2 Failed."

Test for Assertion 16.3

The function is tested using a bad handle (Discovery Method 1, Discovery Mode).

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGcReadTagList().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1603.

3. (Pre) There exists a target container on the connected card with the following properties:
   - the ACR for the gscBsiGcReadTagList() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by AID value ==
     - _goodGSCAID1, of length _goodGSCAID1Len (GSC), or
   - _goodCACAID1, of length _goodCACAID1Len (CAC).

4. (Pre) Print "Testing of Assertion 16.3".

5. (Pre) Establish an authenticated session with the target container on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard1603
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - strctAuthenticator == strctAuthenticator1600
   - unAuthNb == 1.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 6.

   Case 2: If the gscBsiUtilAcquireContext() call does not return the code BSI_OK, then print
   "A session cannot be established. Assertion 16.3 of gscBsiGcDataCreate() cannot be tested".
   End Test for Assertion 16.3.

6. (Pre) Allocate the unsigned long* punNbTags1603 == 0.

7. Make a gscBsiGcReadTagList() call to the SPS, using
   - hCard /= hCard1603
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - tagArray == NULL
   - punNbTags == punNbTags1603.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_BAD_HANDLE or the return code BSI_NO_CARDSERVICE.

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiGcReadTagList() call returns the code BSI_BAD_HANDLE, then:

   Perform Test for Assertion 9.16.3.1 using
   - hCard == hCard1603.

   Print
   "gscBsiGcReadTagList() (Discovery Method 1, Discovery Mode) called with a bad handle has been verified.
   Status: Test 16.3 Passed."

   Case 2: If the gscBsiGcReadTagList() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiGcReadTagList() is not supported.
   Status: Test 16.3 Not Supported."

   Case 3: If the gscBsiGcReadTagList() call does not return the code BSI_BAD_HANDLE or the code BSI_NO_CARDSERVICE, then:

   Print
   "gscBsiGcReadTagList() (Discovery Method 1, Discovery Mode) called with a bad handle returned an incorrect code.
   Status: Test 16.3 Failed."

Test for Assertion 16.4

The function is tested using a bad handle (Discovery Method 1, Final Mode).

Instantiation Scenario:
1. (Pre) Construct the Starting State for the testing of gscBsiGcReadTagList().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1604.

3. (Pre) There exists a target container on the connected card with the following properties:

- the ACR for the gscBsiGcReadTagList() service has the value BSI_ACR_PIN
- the value of the PIN is _PIN
- the container is represented by AID value ==
  - _goodGSCAID1, of length _goodGSCAID1Len (GSC), or
- _goodCACAID1, of length _goodCACAID1Len (CAC).

4. (Pre) Print "Testing of Assertion 16.4".

5. (Pre) Establish an authenticated session with the target container on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard1604
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - strctAuthenticator == strctAuthenticator1600
   - unAuthNb == 1.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 6.

   Case 2: If the gscBsiUtilAcquireContext() call does not return the code BSI_OK, then print
   "A session cannot be established. Assertion 16.4 of gscBsiGcDataCreate() cannot be tested".
   End Test for Assertion 16.4.

6. (Pre) Allocate the unsigned long* punNbTags1604 == 0.

7. Make a gscBsiGcReadTagList() call to the SPS, using
   - hCard == hCard1604
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - tagArray == NULL
   - punNbTags == punNbTags1604.

   Case 1: If the gscBsiGcReadTagList() call returns the code BSI_OK, then print
   "gscBsiGcReadTagList() (Discovery Method 1, Discovery Mode) succeeded."
   Continue with 8.

   Case 2: If the gscBsiGcReadTagList() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiGcReadTagList() is not supported.
   Status: Test 16.4 Not Supported."
   End Test for Assertion 16.4.

   Case 3: If the gscBsiGcReadTagList() call does not return the code BSI_OK or the code BSI_NO_CARDSERVICE, then print
   "gscBsiGcReadTagList() (Discovery Method 1, Discovery Mode) did not return the code BSI_OK.
   Status: Test 16.4 Failed."
   End Test for Assertion 16.4.

8. Allocate the unsigned GCtag array tagArray1604 with length == punNBTags1604*.

9. Make a gscBsiGcReadTagList() call to the SPS, using
   - hCard /= hCard1604
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - tagArray == tagArray1604
   - punNbTags == punNbTags1604.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_BAD_HANDLE or the return code BSI_NO_CARDSERVICE.

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiGcReadTagList() call returns the code BSI_BAD_HANDLE, then:

   Perform Test for Assertion 9.16.4.1 using
   - hCard == hCard1604.

   Print
   "gscBsiGcReadTagList() (Discovery Method 1, Final Mode) called with a bad handle has been verified.
   Status: Test 16.4 Passed."

   Case 2: If the gscBsiGcReadTagList() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiGcReadTagList() is not supported.
   Status: Test 16.4 Not Supported."

   Case 3: If the gscBsiGcReadTagList() call does not return the code BSI_BAD_HANDLE or the code BSI_NO_CARDSERVICE, then:

   Print
   "gscBsiGcReadTagList() (Discovery Method 1, Final Mode) called with a bad handle returned an incorrect code.
   Status: Test 16.4 Failed."

Test for Assertion 16.5

The function is tested with another application having established a transaction lock (Discovery Method 1, Discovery Mode).

Until we encounter implementations that allow multiple simultaneous applications, I don't think we need to worry about this assertion.

Test for Assertion 16.6

The function is tested using a bad AID value (Discovery Method 1, Discovery Mode).

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGcReadTagList().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1606.

3. (Pre) There exists a target container on the connected card with the following properties:
   - the ACR for the gscBsiGcReadTagList() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by AID value ==
     - _goodGSCAID1, of length _goodGSCAID1Len (GSC), or
   - _goodCACAID1, of length _goodCACAID1Len (CAC).

4. (Pre) There does not exist a container on the connected card with AID value ==
   - _badGSCAID, of length _badGSCAIDLen (GSC), or
   - _badCACAID, of length _badCACAIDLen (CAC).

5. (Pre) Print "Testing of Assertion 16.6".

6. (Pre) Establish an authenticated session with the target container on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard1606
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - strctAuthenticator == strctAuthenticator1600
   - unAuthNb == 1.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 8.

   Case 2: If the gscBsiUtilAcquireContext() call does not return the code BSI_OK, then print
   "A session cannot be established. Assertion 16.6 of gscBsiGcDataCreate() cannot be tested".
   End Test for Assertion 16.6.

7. (Pre) Allocate the unsigned long* punNbTags1606 == 0.

8. Make a gscBsiGcReadTagList() call to the SPS, using
   - hCard == hCard1606
   - uszAID == _badGSCAID (GSC) or _badCACAID (CAC)
   - unAIDLen == _badGSCAIDLen (GSC) or _badCACAIDLen (CAC)
   - tagArray == NULL
   - punNbTags == punNbTags1606.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_BAD_AID or the return code BSI_NO_CARDSERVICE.

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiGcReadTagList() call returns the code BSI_BAD_AID, then:

   Perform Test for Assertion 9.16.6.1 using
   - hCard == hCard1606.

   Print
   "gscBsiGcReadTagList() (Discovery Method 1, Discovery Mode) called with a bad AID value has been verified.
   Status: Test 16.6 Passed."

   Case 2: If the gscBsiGcReadTagList() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiGcReadTagList() is not supported.
   Status: Test 16.6 Not Supported."

   Case 3: If the gscBsiGcReadTagList() call does not return the code BSI_BAD_AID or the code BSI_NO_CARDSERVICE, then:

   Print
   "gscBsiGcReadTagList() (Discovery Method 1, Discovery Mode) called with a bad AID value returned an incorrect code.
   Status: Test 16.6 Failed."

Test for Assertion 16.7

The function is tested using a bad AID length (Discovery Method 1, Discovery Mode).

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGcReadTagList().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1607.

3. (Pre) There exists a target container on the connected card with the following properties:
   - the ACR for the gscBsiGcReadTagList() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by AID value ==
     - _goodGSCAID1, of length _goodGSCAID1Len (GSC), or
   - _goodCACAID1, of length _goodCACAID1Len (CAC).

4. (Pre) Print "Testing of Assertion 16.7".

5. (Pre) Establish an authenticated session with the target container on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard1607
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)

- strctAuthenticator == strctAuthenticator1600
- unAuthNb == 1.

Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 7.

Case 2: If the gscBsiUtilAcquireContext() call does not return the code BSI_OK, then print
"A session cannot be established. Assertion 16.7 of gscBsiGcDataCreate() cannot be tested".
End Test for Assertion 16.7.

6. (Pre) Allocate the unsigned long* punNbTags1607 == 0.

7. Make a gscBsiGcReadTagList() call to the SPS, using
   - hCard == hCard1607
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _badAIDLen
   - tagArray == NULL
   - punNbTags == punNbTags1607.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_BAD_PARAM or the return code BSI_NO_CARDSERVICE.

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiGcReadTagList() call returns the code BSI_BAD_PARAM, then:

   Perform Test for Assertion 9.16.7.1 using
   - hCard == hCard1607.

   Print
   "gscBsiGcReadTagList() (Discovery Method 1, Discovery Mode) called with a bad AID length has been verified.
   Status: Test 16.7 Passed."

   Case 2: If the gscBsiGcReadTagList() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiGcReadTagList() is not supported.
   Status: Test 16.7 Not Supported."

   Case 3: If the gscBsiGcReadTagList() call does not return the code BSI_BAD_PARAM or the code BSI_NO_CARDSERVICE, then:

   Print
   "gscBsiGcReadTagList() (Discovery Method 1, Discovery Mode) called with a bad AID length returned an incorrect code.
   Status: Test 16.7 Failed."

Test for Assertion 16.8

The function is tested with a removed card (Discovery Method 1, Discovery Mode).

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGcReadTagList().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1608.

3. (Pre) There exists a target container on the connected card with the following properties:
   - the ACR for the gscBsiGcReadTagList() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by AID value ==
     - _goodGSCAID1, of length _goodGSCAID1Len (GSC), or
   - _goodCACAID1, of length _goodCACAID1Len (CAC).

4. (Pre) Print "Testing of Assertion 16.8".

5. (Pre) Establish an authenticated session with the target container on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard1608
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - strctAuthenticator == strctAuthenticator1600
   - unAuthNb == 1.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 6.

   Case 2: If the gscBsiUtilAcquireContext() call does not return the code BSI_OK, then print
   "A session cannot be established. Assertion 16.8 of gscBsiGcReadTagList() cannot be tested".
   End Test for Assertion 16.8.

6. (Pre) Allocate the unsigned long* punNbTags1608 == 0.

7. Remove the connected card from the reader.

8. Make a gscBsiGcReadTagList() call to the SPS, using
   - hCard == hCard1608
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - tagArray == NULL
   - punNbTags == punNbTags1608.

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_CARD_REMOVED or the return code BSI_NO_CARDSERVICE.

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiGcReadTagList() call returns the code BSI_CARD_REMOVED, then:

   Print
   "gscBsiGcReadTagList() (Discovery Method 1, Discovery Mode) called with the connected card removed has been verified.
   Status: Test 16.8 Passed."

   Case 2: If the gscBsiGcReadTagList() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiGcReadTagList() is not supported.
   Status: Test 16.8 Not Supported."

   Case 3: If the gscBsiGcReadTagList() call does not return the code BSI_CARD_REMOVED or the code BSI_NO_CARDSERVICE, then:

   Print
   "gscBsiGcReadTagList() (Discovery Method 1, Discovery Mode) called with the connected card removed returned an incorrect code.
   Status: Test 16.8 Failed."

Test for Assertion 16.9

The function is tested without fulfilling the applicable ACR (Discovery Method 1, Discovery Mode).

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGcReadTagList().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1609.

3. (Pre) There exists a target container on the connected card with the following properties:
   - the ACR for the gscBsiGcReadTagList() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by AID value ==
     - _goodGSCAID1, of length _goodGSCAID1Len (GSC), or
   - _goodCACAID1, of length _goodCACAID1Len (CAC).

4. (Pre) Print "Testing of Assertion 16.9".

5. (Pre) Ensure that there is no authenticated session with the target container:

   Make a gscBsiUtilReleaseContext() call to the SPS, using
   - hCard == hCard1609
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC).

6. (Pre) Allocate the unsigned long* punNbTags1609 == 0.

7. Make a gscBsiGcReadTagList() call to the SPS, using
   - hCard == hCard1609
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - tagArray == NULL
   - punNbTags == punNbTags1609.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_ACCESS_DENIED or the return code BSI_NO_CARDSERVICE.

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiGcReadTagList() call returns the code BSI_ACCESS_DENIED, then:

   Perform Test for Assertion 9.16.9.1 using
   - hCard == hCard1609.

   Print
   "gscBsiGcReadTagList() (Discovery Method 1, Discovery Mode) called without fulfilling the applicable ACR has been verified.
   Status: Test 16.9 Passed."

   Case 2: If the gscBsiGcReadTagList() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiGcReadTagList() is not supported.
   Status: Test 16.9 Not Supported."

   Case 3: If the gscBsiGcReadTagList() call does not return the code BSI_ACCESS_DENIED or the code BSI_NO_CARDSERVICE, then print
   "gscBsiGcReadTagList() (Discovery Method 1, Discovery Mode) called without fulfilling the applicable ACR returned an incorrect code.
   Status: Test 16.9 Failed."

17. gscBsiGcReadValue()

Starting State for Each Test:

1. There exists a BSIAuthenticator array strctAuthenticator1700 with one element, the structure BSIAuthenticator1700. This structure has fields
   - unAccessMethodType1700 == BSI_AM_PIN
   - unkeyIDOrReference1700 == _keyIDOrReference1
   - uszAuthValue1700 == _goodAuthValue1
   - unAuthValueLen1700 == _goodAuthValue1Len.

Test for Assertion 17.1

The function is tested using valid parameters (Discovery Method 1).

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGcReadValue().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1701.

3. (Pre) There exists a target container on the connected card with the following properties:
   - the ACR for the gscBsiGcReadValue() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by AID value ==
     - _goodGSCAID1, of length _goodGSCAID1Len (GSC), or
     - _goodCACAID1, of length _goodCACAID1Len (CAC)
   - the container contains a data item for which
     - the tag is == _existingTag1701
     - the value is == existingDvalue1701.

4. (Pre) Print "Testing of Assertion 17.1".

5. (Pre) Establish an authenticated session with the target container on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard1701
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - strctAuthenticator == strctAuthenticator1700
   - unAuthNb == 1.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 6.

   Case 2: If the gscBsiUtilAcquireContext() call does not return the code BSI_OK, then print
   "A session cannot be established. Assertion 17.1 of gscBsiGcDataCreate() cannot be tested".
   End Test for Assertion 17.1.

6. (Pre) Allocate the unsigned long* punValueLen1701 == 0.

7. Make a gscBsiGcReadValue() call to the SPS, using
   - hCard == hCard1701
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - GCtag == _existingTag1701
   - uszValue == NULL
   - punValueLen == punValueLen1701.

   Case 1: If the gscBsiGcReadValue() call returns the code BSI_OK, then print
   "gscBsiGcReadValue() (Discovery Method 1, Discovery Mode) called with valid parameters succeeded."
   Continue with 8.

   Case 2: If the gscBsiGcReadValue() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiGcReadTagList() is not supported.
   Status: Test 17.1 Not Supported."
   End Test for Assertion 17.1

   Case 3: If the gscBsiGcReadValue() call does not return the code BSI_OK or the code BSI_NO_CARDSERVICE, then print
   "gscBsiGcReadValue() (Discovery Method 1, Discovery Mode) called with valid parameters returned an incorrect code.
   Status: Test 17.1 Failed."
   End Test for Assertion 17.1.

8. Allocate the unsigned string uszValue1701 with length == punValueLen1701*.

9. Make a gscBsiGcReadValue() call to the SPS, using
   - hCard == hCard1701
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - GCtag == _existingTag1701
   - uszValue == uszValue1701
   - punValueLen == punValueLen1701.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_OK or the return code BSI_NO_CARDSERVICE.

2. If the return code is BSI_OK, then uszValue1701 == _existingDvalue1701.

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiGcReadValue() call returns the code BSI_OK, then:

   Case 1.1: If uszValue1701 == _existingDvalue1701, then print
   "gscBsiGcReadValue() (Discovery Method 1, Final Mode) called with valid parameters has been verified.
   Status: Test 17.1 Passed."

Case 1.2: If uszValue1701 /= _existingDvalue1701, then print
"gscBsiGcReadValue() (Discovery Method 1, Final Mode) called with valid parameters has not been verified.
Status: Test 17.1 Failed."

Case 2: If the gscBsiGcReadValue() call returns the code BSI_NO_CARDSERVICE, then print
"gscBsiGcReadValue() is not supported.
Status: Test 17.1 Not Supported."

Case 3: If the gscBsiGcReadValue() call does not return the code BSI_OK or the code BSI_NO_CARDSERVICE, then print
"gscBsiGcReadValue() (Discovery Method 1, Final Mode) called with valid parameters returned an incorrect code.
Status: Test 17.1 Failed."

Test for Assertion 17.2

The function is tested using valid parameters (Discovery Method 2).

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGcReadValue().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1702.

3. (Pre) There exists a target container on the connected card with the following properties:
   - the ACR for the gscBsiGcReadValue() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by AID value ==
     - _goodGSCAID1, of length _goodGSCAID1Len (GSC), or
   - _goodCACAID1, of length _goodCACAID1Len (CAC)
   - the container contains a data item for which
     - the tag is == _existingTag1702
     - the value is == existingDvalue1702.

4. (Pre) Print "Testing of Assertion 17.2".

5. (Pre) Establish an authenticated session with the target container on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard1702
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - strctAuthenticator == strctAuthenticator1700
   - unAuthNb == 1.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 6.

Case 2: If the gscBsiUtilAcquireContext() call does not return the code BSI_OK, then print
"A session cannot be established. Assertion 17.2 of gscBsiGcDataCreate() cannot be tested".
End Test for Assertion 17.2.

6. (Pre) Allocate the unsigned string uszValue1702 with length == 0.

7. (Pre) Allocate the unsigned long* punValueLen1702 == 0.

8. Make a gscBsiGcReadValue() call to the SPS, using
   - hCard == hCard1702
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - GCtag == _existingTag1702
   - uszValue == uszValue1702
   - punValueLen == punValueLen1702.

   Case 1: If the gscBsiGcReadValue() call returns the code BSI_INSUFFICIENT_BUFFER, then print
   "gscBsiGcReadValue() (Discovery Method 2, Discovery Mode) correctly returned the code BSI_INSUFFICIENT_BUFFER."
   Continue with 9.

   Case 2: If the gscBsiGcReadValue() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiGcReadValue() is not supported.
   Status: Test 17.2 Not Supported."
   End Test for Assertion 17.2.

   Case 3: If the gscBsiGcReadValue() call does not return the code BSI_INSUFFICIENT_BUFFER or the code BSI_NO_CARDSERVICE, then print
   "gscBsiGcReadValue() (Discovery Method 2, Discovery Mode) returned an incorrect code.
   Status: Test 17.2 Failed."
   End Test for Assertion 17.2.

9. Re-allocate the unsigned string uszValue1702 with length == punValueLen1702*.

10. Make a gscBsiGcReadValue() call to the SPS, using
    - hCard == hCard1702
    - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
    - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
    - GCtag == _existingTag1702
    - uszValue == uszValue1702
    - punValueLen == punValueLen1702.

<u>Verification Goal</u>:
To verify the Expected Results:

1. The call returns
   - the return code BSI_OK or the return code BSI_NO_CARDSERVICE.

2. If the return code is BSI_OK, then uszValue1702 == _existingDvalue1702.

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiGcReadValue() call returns the code BSI_OK, then:

   Case 1.1: If uszValue1702 == _existingDvalue1702, then print "gscBsiGcReadValue() (Discovery Method 2, Final Mode) called with valid parameters has been verified.
   Status: Test 17.2 Passed."

   Case 1.2: If uszValue1702 /= _existingDvalue1702, then print "gscBsiGcReadValue() (Discovery Method 2, Final Mode) called with valid parameters has not been verified.
   Status: Test 17.2 Failed."

   Case 2: If the gscBsiGcReadValue() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiGcReadValue() is not supported.
   Status: Test 17.2 Not Supported."

   Case 3: If the gscBsiGcReadValue() call does not return the code BSI_OK or the code BSI_NO_CARDSERVICE, then print
   "gscBsiGcReadValue() (Discovery Method 2, Final Mode) called with valid parameters returned an incorrect code.
   Status: Test 17.2 Failed."

Test for Assertion 17.3

The function is tested using a bad handle (Discovery Method 1, Discovery Mode).

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGcReadValue().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1703.

3. (Pre) There exists a target container on the connected card with the following properties:
   - the ACR for the gscBsiGcReadValue() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by AID value ==
     - _goodGSCAID1, of length _goodGSCAID1Len (GSC), or
     - _goodCACAID1, of length _goodCACAID1Len (CAC)
   - the container contains a data item for which
     - the tag is == _existingTag1703.

4. (Pre) Print "Testing of Assertion 17.3".

5. (Pre) Establish an authenticated session with the target container on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard1703
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)

- unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
- strctAuthenticator == strctAuthenticator1700
- unAuthNb == 1.

Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 6.

Case 2: If the gscBsiUtilAcquireContext() call does not return the code BSI_OK, then print
"A session cannot be established. Assertion 17.3 of gscBsiGcDataCreate() cannot be tested".
End Test for Assertion 17.3.

6. (Pre) Allocate the unsigned long* punValueLen1703 == 0.

7. Make a gscBsiGcReadValue() call to the SPS, using
   - hCard /= hCard1703
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - GCtag == _existingTag1703
   - uszValue == NULL
   - punValueLen == punValueLen1703.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_BAD_HANDLE or the return code BSI_NO_CARDSERVICE.

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiGcReadValue() call returns the code BSI_BAD_HANDLE, then:

   Perform Test for Assertion 9.17.3.1 using
   - hCard == hCard1703.

   Print
   "gscBsiGcReadValue() (Discovery Method 1, Discovery Mode) called with a bad handle has been verified."
   Status: Test 17.3 Passed."

   Case 2: If the gscBsiGcReadValue() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiGcReadValue() is not supported.
   Status: Test 17.3 Not Supported."

   Case 3: If the gscBsiGcReadValue() call does not return the code BSI_BAD_HANDLE or the code BSI_NO_CARDSERVICE, then:

   Print
   "gscBsiGcReadValue() (Discovery Method 1, Discovery Mode) called with a bad handle returned an incorrect code.
   Status: Test 17.3 Failed."

Test for Assertion 17.4

The function is tested using a bad handle (Discovery Method 2, Final Mode).

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGcReadValue().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1704.

3. (Pre) There exists a target container on the connected card with the following properties:
    - the ACR for the gscBsiGcReadValue() service has the value BSI_ACR_PIN
    - the value of the PIN is _PIN
    - the container is represented by AID value ==
        - _goodGSCAID1, of length _goodGSCAID1Len (GSC), or
        - _goodCACAID1, of length _goodCACAID1Len (CAC)
    - the container contains a data item for which
        - the tag is == _existingTag1704.

4. (Pre) Print "Testing of Assertion 17.4".

5. (Pre) Establish an authenticated session with the target container on the card:

    Make a gscBsiUtilAcquireContext() call to the SPS, using
    - hCard == hCard1704
    - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
    - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
    - strctAuthenticator == strctAuthenticator1700
    - unAuthNb == 1.

    Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 6.

    Case 2: If the gscBsiUtilAcquireContext() call does not return the code BSI_OK, then print
    "A session cannot be established. Assertion 17.4 of gscBsiGcDataCreate() cannot be tested".
    End Test for Assertion 17.4.

6. (Pre) Allocate the unsigned string uszValue1704 with length == 0.

7. (Pre) Allocate the unsigned long* punValueLen1704 == 0.

8. Make a gscBsiGcReadValue() call to the SPS, using
    - hCard == hCard1704
    - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
    - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
    - GCtag == _existingTag1704
    - uszValue == uszValue1704
    - punValueLen == punValueLen1704.

Case 1: If the gscBsiGcReadValue() call returns the code BSI_INSUFFICIENT_BUFFER, then print
"gscBsiGcReadValue() (Discovery Method 1, Discovery Mode) called with valid parameters correctly returned BSI_INSUFFICIENT_BUFFER."
Continue with 9.

Case 2: If the gscBsiGcReadValue() call returns the code BSI_NO_CARDSERVICE, then print
"gscBsiGcReadValue() is not supported.
Status: Test 17.4 Not Supported."

Case 3: If the gscBsiGcReadValue() call does not return the code BSI_INSUFFICIENT_BUFFER or the code BSI_NO_CARDSERVICE, then print
"gscBsiGcReadValue() (Discovery Method 1, Discovery Mode) called with valid parameters returned an incorrect code.
Status: Test 17.4 Failed."
End Test for Assertion 17.4.

9. Re-allocate the unsigned string uszValue1704 with length == punValueLen1704*.

10. Make a gscBsiGcReadValue() call to the SPS, using
    - hCard /= hCard1704
    - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
    - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
    - GCtag == _existingTag1704
    - uszValue == uszValue1704
    - punValueLen == punValueLen1704.

Verification Goal:
To verify the Expected Results:
1. The call returns
    - the return code BSI_BAD_HANDLE or the return code BSI_NO_CARDSERVICE.

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiGcReadValue() call returns the code BSI_BAD_HANDLE, then:

   Perform Test for Assertion 9.17.4.1 using
   - hCard == hCard1704.

   Print
   "gscBsiGcReadValue() (Discovery Method 1, Final Mode) called with a bad handle has been verified."
   Status: Test 17.4 Passed."

   Case 2: If the gscBsiGcReadValue() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiGcReadValue() is not supported.
   Status: Test 17.4 Not Supported."

   Case 3: If the gscBsiGcReadValue() call does not return the code BSI_BAD_HANDLE or the code BSI_NO_CARDSERVICE, then:

Print
"gscBsiGcReadValue() (Discovery Method 1, Final Mode) called with a bad handle returned an incorrect code.
Status: Test 17.4 Failed."

Test for Assertion 17.5

The function is tested with another application having established a transaction lock (Discovery Method 1, Discovery Mode).

Until we encounter implementations that allow multiple simultaneous applications, I don't think we need to worry about this assertion.

Test for Assertion 17.6

The function is tested using a bad AID value (Discovery Method 1, Discovery Mode).

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGcReadValue().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1706.

3. (Pre) There exists a target container on the connected card with the following properties:
   - the ACR for the gscBsiGcReadValue() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by AID value ==
     - _goodGSCAID1, of length _goodGSCAID1Len (GSC), or
     - _goodCACAID1, of length _goodCACAID1Len (CAC)
   - the container contains a data item for which
     - the tag is == _existingTag1706.

4. (Pre) There does not exist a container on the card with AID value ==
   - _badGSCAID, of length _badGSCAIDLen (GSC), or
   - _badCACAID, of length _badCACAIDLen (CAC).

5. (Pre) Print "Testing of Assertion 17.6".

6. (Pre) Establish an authenticated session with the target container on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard1706
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - strctAuthenticator == strctAuthenticator1700
   - unAuthNb == 1.

Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 8.

Case 2: If the gscBsiUtilAcquireContext() call does not return the code BSI_OK, then print
"A session cannot be established. Assertion 17.6 of gscBsiGcDataCreate() cannot be tested".
End Test for Assertion 17.6.

7. (Pre) Allocate the unsigned long* punValueLen1706 == 0.

8. Make a gscBsiGcReadValue() call to the SPS, using
   - hCard == hCard1706
   - uszAID == _badGSCAID (GSC) or _badCACAID (CAC)
   - unAIDLen == _badGSCAIDLen (GSC) or _badCACAIDLen (CAC)
   - GCtag == _existingTag1706
   - uszValue == NULL
   - punValueLen == punValueLen1706.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_BAD_AID or the return code BSI_NO_CARDSERVICE.

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiGcReadValue() call returns the code BSI_BAD_AID, then:

   Perform Test for Assertion 9.17.6.1 using
   - hCard == hCard1706.

   Print
   "gscBsiGcReadValue() (Discovery Method 1, Discovery Mode) called with a bad AID value has been verified."
   Status: Test 17.6 Passed."

   Case 2: If the gscBsiGcReadValue() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiGcReadValue() is not supported.
   Status: Test 17.6 Not Supported."

   Case 3: If the gscBsiGcReadValue() call does not return the code BSI_BAD_AID or the code BSI_NO_CARDSERVICE, then:

   Print
   "gscBsiGcReadValue() (Discovery Method 1, Discovery Mode) called with a bad AID value returned an incorrect code.
   Status: Test 17.6 Failed."

Test for Assertion 17.7

The function is tested using a bad AID length (Discovery Method 1, Discovery Mode).

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGcReadValue().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1707.

3. (Pre) There exists a target container on the connected card with the following properties:
   - the ACR for the gscBsiGcReadValue() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by AID value ==
     - _goodGSCAID1, of length _goodGSCAID1Len (GSC), or
     - _goodCACAID1, of length _goodCACAID1Len (CAC)
   - the container contains a data item for which
     - the tag is == _existingTag1707.

4. (Pre) Print "Testing of Assertion 17.7".

5. (Pre) Establish an authenticated session with the target container on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard1707
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - strctAuthenticator == strctAuthenticator1700
   - unAuthNb == 1.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 7.

   Case 2: If the gscBsiUtilAcquireContext() call does not return the code BSI_OK, then print
   "A session cannot be established. Assertion 17.7 of gscBsiGcDataCreate() cannot be tested".
   End Test for Assertion 17.7.

6. (Pre) Allocate the unsigned long* punValueLen1707 == 0.

7. Make a gscBsiGcReadValue() call to the SPS, using
   - hCard == hCard1707
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _badAIDLen
   - GCtag == _existingTag1707
   - uszValue == NULL
   - punValueLen == punValueLen1707.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_BAD_PARAM or the return code BSI_NO_CARDSERVICE.

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiGcReadValue() call returns the code BSI_BAD_PARAM, then:

   Perform Test for Assertion 9.17.7.1 using
   - hCard == hCard1707.

   Print
   "gscBsiGcReadValue() (Discovery Method 1, Discovery Mode) called with a bad AID length has been verified."
   Status: Test 17.7 Passed."

   Case 2: If the gscBsiGcReadValue() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiGcReadValue() is not supported.
   Status: Test 17.7 Not Supported."

   Case 3: If the gscBsiGcReadValue() call does not return the code BSI_BAD_PARAM or the code BSI_NO_CARDSERVICE, then:

   Print
   "gscBsiGcReadValue() (Discovery Method 1, Discovery Mode) called with a bad AID length returned an incorrect code.
   Status: Test 17.7 Failed."

Test for Assertion 17.8

The function is tested using a bad tag (Discovery Method 1, Discovery Mode).

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGcReadValue().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1708.

3. (Pre) There exists a target container on the connected card with the following properties:
   - the ACR for the gscBsiGcReadValue() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by AID value ==
     - _goodGSCAID1, of length _goodGSCAID1Len (GSC), or
     - _goodCACAID1, of length _goodCACAID1Len (CAC)
   - the container does not contain a data item for which the tag == _newTag1708.

4. (Pre) Print "Testing of Assertion 17.8".

5. (Pre) Establish an authenticated session with the target container on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard1708
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)

- unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
- strctAuthenticator == strctAuthenticator1700
- unAuthNb == 1.

Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 6.

Case 2: If the gscBsiUtilAcquireContext() call does not return the code BSI_OK, then print
"A session cannot be established. Assertion 17.8 of gscBsiGcDataCreate() cannot be tested".
End Test for Assertion 17.8.

6. (Pre) Allocate the unsigned long* punValueLen1708 == 0.

7. Make a gscBsiGcReadValue() call to the SPS, using
   - hCard == hCard1708
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - GCtag == _newTag1708
   - uszValue == NULL
   - punValueLen == punValueLen1708.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_BAD_TAG or the return code BSI_NO_CARDSERVICE.

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiGcReadValue() call returns the code BSI_BAD_TAG, then:

   Perform Test for Assertion 9.17.8.1 using
   - hCard == hCard1708.

   Print
   "gscBsiGcReadValue() (Discovery Method 1, Discovery Mode) called with a bad tag has been verified."
   Status: Test 17.8 Passed."

   Case 2: If the gscBsiGcReadValue() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiGcReadValue() is not supported.
   Status: Test 17.8 Not Supported."

   Case 3: If the gscBsiGcReadValue() call does not return the code BSI_BAD_TAG or the code BSI_NO_CARDSERVICE, then:

   Print
   "gscBsiGcReadValue() (Discovery Method 1, Discovery Mode) called with a bad tag returned an incorrect code.
   Status: Test 17.8 Failed."

Test for Assertion 17.9

The function is tested with a removed card (Discovery Method 1, Discovery Mode).

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGcReadValue().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1709.

3. (Pre) There exists a target container on the connected card with the following properties:
   - the ACR for the gscBsiGcReadValue() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by AID value ==
     - _goodGSCAID1, of length _goodGSCAID1Len (GSC), or
     - _goodCACAID1, of length _goodCACAID1Len (CAC)
   - the container contains a data item for which
     - the tag is == _existingTag1709.

4. (Pre) Print "Testing of Assertion 17.9".

5. (Pre) Establish an authenticated session with the target container on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard1709
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - strctAuthenticator == strctAuthenticator1700
   - unAuthNb == 1.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 6.

   Case 2: If the gscBsiUtilAcquireContext() call does not return the code BSI_OK, then print
   "A session cannot be established. Assertion 17.9 of gscBsiGcDataCreate() cannot be tested".
   End Test for Assertion 17.9.

6. (Pre) Allocate the unsigned long* punValueLen1709 == 0.

7. (Pre) Remove the connected card from the reader.

8. Make a gscBsiGcReadValue() call to the SPS, using
   - hCard == hCard1709
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - GCtag == _existingTag1709
   - uszValue == NULL
   - punValueLen == punValueLen1709.

Verification Goal:

To verify the Expected Results:

1. The call returns
   - the return code BSI_CARD_REMOVED or the return code BSI_NO_CARDSERVICE.

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiGcReadValue() call returns the code BSI_CARD_REMOVED, then:

   Print
   "gscBsiGcReadValue()(Discovery Method 1, Discovery Mode) called with the connected card removed has been verified."
   Status: Test 17.9 Passed."

   Case 2: If the gscBsiGcReadValue() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiGcReadValue() is not supported.
   Status: Test 17.9 Not Supported."

   Case 3: If the gscBsiGcReadValue() call does not return the code BSI_CARD_REMOVED or the code BSI_NO_CARDSERVICE, then:

   Print
   "gscBsiGcReadValue() (Discovery Method 1, Discovery Mode) called with the connected card removed returned an incorrect code.
   Status: Test 17.9 Failed."

Test for Assertion 17.10

The function is tested without fulfilling the applicable ACR (Discovery Method 1, Discovery Mode).

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGcReadValue().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1710.

3. (Pre) There exists a target container on the connected card with the following properties:
   - the ACR for the gscBsiGcReadValue() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by AID value ==
     - _goodGSCAID1, of length _goodGSCAID1Len (GSC), or
     - _goodCACAID1, of length _goodCACAID1Len (CAC)
   - the container contains a data item for which
     - the tag is == _existingTag1710.

4. (Pre) Print "Testing of Assertion 17.10".

5. Ensure that there is no an authenticated session with the target container:

Make a gscBsiUtilReleaseContext() call to the SPS, using
- hCard == hCard1710
- uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
- unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC).

6. (Pre) Allocate the unsigned long* punValueLen1710 == 0.

7. Make a gscBsiGcReadValue() call to the SPS, using
   - hCard == hCard1710
   - uszAID == _goodGSCAID2 (GSC) or _goodCACAID2 (CAC)
   - unAIDLen == _goodGSCAID2Len (GSC) or _goodCACAID2Len (CAC)
   - GCtag == _existingTag1710
   - uszValue == NULL
   - punValueLen == punValueLen1710.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_ACCESS_DENIED or the return code BSI_NO_CARDSERVICE.

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiGcReadValue() call returns the code BSI_ACCESS_DENIED, then:

   Perform Test for Assertion 9.17.10.1 using
   - hCard == hCard1710.

   Print
   "gscBsiGcReadValue() (Discovery Method 1, Discovery Mode) called without fulfilling the applicable ACR has been verified."
   Status: Test 17.10 Passed."

   Case 2: If the gscBsiGcReadValue() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiGcReadValue() is not supported.
   Status: Test 17.10 Not Supported."

   Case 3: If the gscBsiGcReadValue() call does not return the code BSI_ACCESS_DENIED or the code BSI_NO_CARDSERVICE, then:

   Print
   "gscBsiGcReadValue() (Discovery Method 1, Discovery Mode) called without fulfilling the applicable ACR returned an incorrect code.
   Status: Test 17.10 Failed."

18. gscBsiGcUpdateValue()

Starting State for Each Test:

1. There exists a BSIAuthenticator array strctAuthenticator1800 with one element, the structure BSIAuthenticator1800. This structure has fields
   - unAccessMethodType1800 == BSI_AM_PIN
   - unkeyIDOrReference1800 == _keyIDOrReference1
   - uszAuthValue1800 == _goodAuthValue1
   - unAuthValueLen1800 == _goodAuthValue1Len.

Test for Assertion 18.1

The function is tested using valid parameters.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGcUpdateValue().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1801.

3. (Pre) There exists a target container on the connected card with the following properties:
   - the ACR for the gscBsiGcUpdateValue() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by AID value ==
     - _goodGSCAID1, of length _goodGSCAID1Len (GSC), or
     - _goodCACAID1, of length _goodCACAID1Len (CAC)
   - the container contains a data item for which
     - the tag is == _existingTag1801
     - the value is == _existingDvalue1801.

4. (Pre) Print "Testing of Assertion 18.1".

5. (Pre) Establish an authenticated session with the target container on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard1801
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - strctAuthenticator == strctAuthenticator1800
   - unAuthNb == 1.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 6.

   Case 2: If the gscBsiUtilAcquireContext() call does not return the code BSI_OK, then print
   "A session cannot be established. Assertion 18.1 of gscBsiGcUpdateValue() cannot be tested".
   End Test for Assertion 18.1.

6. Make a gscBsiGcUpdateValue() call to the SPS, using
    - hCard == hCard1801
    - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
    - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
    - ucTag == _existingTag1801
    - uszValue == _newDvalue1801
    - unValueLen == _newDvalueLen.

Verification Goal:
To verify the Expected Results:

1. The call returns
    - the return code BSI_OK or the return code BSI_NO_CARDSERVICE.

2. If the return code is BSI_OK, then the data item on the target container for which the tag is _existingTag1801 now has the value _newDvalue1801.

3. No other changes are made to the container structure of the connected card.

Note: I suppose if we were very ambitious, we would verify 3., by comparing before and after snapshots of every container on the connected card. For now, I propose limiting ourselves to verifying that the specified data was deleted from the selected container.

Perform this verification by issuing a call to gscBsiGcReadValue().

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiGcUpdateValue() call returns the code BSI_OK, then verify that that the specified data item now has the value _newDvalue1801.

    Allocate the unsigned char* uszValue1801 with length _newDvalueLen.

    Make a gscBsiGcReadValue() call to the SPS, using
    - hCard == hCard1801
    - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
    - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
    - ucTag == _existingTag1801
    - uszValue == uszValue1801
    - punValueLen == _newDvalueLen.

    Case 1.1: If the gscBsiGcReadValue() call returns
    - the code BSI_OK
    - uszValue1801 == _newDvalue1801

    then:

    Print
    "gscBsiGcUpdateValue() called with valid parameters has been verified because a subsequent call to gscBsiGcReadValue() found that the data item had been correctly updated.
    Status: Test 18.1 Passed."

Case 1.2: If the gscBsiReadValue() call returns
- the code BSI_OK
- uszValue1801 /= _newDvalue1801

then:

Print
"gscBsiGcUpdateValue() called with valid parameters has not been verified because a subsequent call to gscBsiGcReadValue() found that the data item was not updated correctly.
Status: Test 18.1 Failed."

Case 1.3: If the gscBsiGcReadValue() call returns any code other than
- BSI_OK

then:

Print
"gscBsiGcUpdateValue() called with valid parameters has not been verified because a subsequent call to gscBsiGcReadValue() was ambiguous.
Status: Test 18.1 Undetermined."

Case 2: If the gscBsiGcUpdateValue() call returns the code BSI_NO_CARDSERVICE, then print
"gscBsiGcUpdateValue() is not supported.
Status: Test 18.1 Not Supported."

Case 3: If the gscBsiGcUpdateValue() call does not return the code BSI_OK or the code BSI_NO_CARDSERVICE, then:

Print
"gscBsiGcUpdateValue() called with valid parameters returned an incorrect code.
Status: Test 18.1 Failed."

Test for Assertion 18.2

The function is tested using a bad handle.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGcUpdateValue().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1802.

3. (Pre) There exists a target container on the connected card with the following properties:
   - the ACR for the gscBsiGcUpdateValue() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by AID value ==
     - _goodGSCAID1, of length _goodGSCAID1Len (GSC), or
     - _goodCACAID1, of length _goodCACAID1Len (CAC)
   - the container contains a data item for which

- the tag is == _existingTag1802
- the value is == _existingDvalue1802.

4. (Pre) Print "Testing of Assertion 18.2".

5. (Pre) Establish an authenticated session with the target container on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard1802
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - strctAuthenticator == strctAuthenticator1800
   - unAuthNb == 1.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 6.

   Case 2: If the gscBsiUtilAcquireContext() call does not return the code BSI_OK, then:

   print
   "A session cannot be established. Assertion 18.2 of gscBsiGcUpdateValue() cannot be tested".
   End Test for Assertion 18.2.

6. Make a gscBsiGcUpdateValue() call to the SPS, using
   - hCard /= hCard1802
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - ucTag == _existingTag1802
   - uszValue == _newDvalue1802
   - unValueLen == _newDvalueLen.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_BAD_HANDLE or the return code BSI_NO_CARDSERVICE.

2. No changes are made to the container structure of the connected card.

Perform this verification by issuing a call to gscBsiGcReadValue().

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiGcUpdateValue() call returns the code BSI_BAD_HANDLE, then:

   Perform Test for Assertion 9.18.2.1 using
   - hCard == hCard1802.

   Allocate the unsigned char* uszValue1801 with length _newDvalueLen.

   Make a gscBsiGcReadValue() call to the SPS, using

- hCard == hCard1802
- uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
- unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
- ucTag == _existingTag1802
- uszValue == uszValue1802
- punValueLen == _newDvalueLen.

Case 1.1: If the gscBsiGcReadValue() call returns
- the code BSI_OK
- uszValue1802 == _existingDvalue1802

then:

Print
"gscBsiGcUpdateValue() called with a bad handle has been verified because a subsequent call to gscBsiGcReadValue() found the original data item.
Status: Test 18.2 Passed."

Case 1.2: If the gscBsiReadValue() call returns
- the code BSI_OK
- uszValue1802 /= _existingDvalue1802

then:

Print
"gscBsiGcUpdateValue() called with a bad handle has not been verified because a subsequent call to gscBsiGcReadValue() found that the data item had been updated or otherwise changed.
Status: Test 18.2 Failed."

Case 1.3: If the gscBsiGcReadValue() call returns any code other than
- BSI_OK

then:

Print
"gscBsiGcUpdateValue() called with a bad handle has not been verified because a subsequent call to gscBsiGcReadValue() was ambiguous.
Status: Test 18.2 Undetermined."

Case 2: If the gscBsiGcUpdateValue() call returns the code BSI_NO_CARDSERVICE, then print
"gscBsiGcUpdateValue() is not supported.
Status: Test 18.2 Not Supported."

Case 3: If the gscBsiGcUpdateValue() call does not return the code BSI_BAD_HANDLE or the code BSI_NO_CARDSERVICE, then:

Print
"gscBsiGcUpdateValue() called with a bad handle returned an incorrect code.
Status: Test 18.2 Failed."

Test for Assertion 18.3

The function is tested with another application having established a transaction lock.

Until we encounter implementations that allow multiple simultaneous applications, I don't think we need to worry about this assertion.

Test for Assertion 18.4

The function is tested using a bad AID value.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGcUpdateValue().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1804.

3. (Pre) There exists a target container on the connected card with the following properties:
   - the ACR for the gscBsiGcUpdateValue() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by AID value ==
     - _goodGSCAID1, of length _goodGSCAID1Len (GSC), or
     - _goodCACAID1, of length _goodCACAID1Len (CAC)
   - the container contains a data item for which
     - the tag is == _existingTag1804
     - the value is == _existingDvalue1804.

4. (Pre) There does not exist a container on the connected card with AID value ==
   - _badGSCAID, of length _badGSCAIDLen (GSC), or
     - _badCACAID, of length _badCACAIDLen (CAC).

5. (Pre) Print "Testing of Assertion 18.4".

6. (Pre) Establish an authenticated session with the target container on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard1804
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - strctAuthenticator == strctAuthenticator1800
   - unAuthNb == 1.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 7.

   Case 2: If the gscBsiUtilAcquireContext() call does not return the code BSI_OK, then print
   "A session cannot be established. Assertion 18.4 of gscBsiGcUpdateValue() cannot be tested".
   End Test for Assertion 18.4.

7. Make a gscBsiGcUpdateValue() call to the SPS, using
    - hCard == hCard1804
    - uszAID == _badGSCAID (GSC) or _badCACAID (CAC)
    - unAIDLen == _badGSCAIDLen (GSC) or _badCACAIDLen (CAC)
    - ucTag == _existingTag1804
    - uszValue == _newDvalue1804
    - unValueLen == _newDvalueLen.

Verification Goal:
To verify the Expected Results:
1. The call returns
    - the return code BSI_BAD_AID or the return code BSI_NO_CARDSERVICE.

2. No changes are made to the container structure of the connected card.

Perform this verification by issuing a call to gscBsiGcReadValue().

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiGcUpdateValue() call returns the code BSI_BAD_AID, then:

    Perform Test for Assertion 9.18.4.1 using
    - hCard == hCard1804.

    Allocate the unsigned char* uszValue1804 with length _newDvalueLen.

    Make a gscBsiGcReadValue() call to the SPS, using
    - hCard == hCard1804
    - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
    - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
    - ucTag == _existingTag1804
    - uszValue == uszValue1804
    - punValueLen == _newDvalueLen.

    Case 1.1: If the gscBsiGcReadValue() call returns
    - the code BSI_OK
    - uszValue1804 == _existingDvalue1804

    then:

    Print "gscBsiGcUpdateValue() called with a bad AID value has been verified because a subsequent call to gscBsiGcReadValue() found the original data item. **Status**: Test 18.4 Passed."

    Case 1.2: If the gscBsiReadValue() call returns
    - the code BSI_OK
    - uszValue1804 /= _existingDvalue1804

    then:

    Print

"gscBsiGcUpdateValue() called with a bad AID value has not been verified because a subsequent call to gscBsiGcReadValue() found that the data item had been updated or otherwise changed.
Status: Test 18.4 Failed."

Case 1.3: If the gscBsiGcReadValue() call returns any code other than
- BSI_OK

then:

Print
"gscBsiGcUpdateValue() called with a bad AID value has not been verified because a subsequent call to gscBsiGcReadValue() was ambiguous.
Status: Test 18.4 Undetermined."

Case 2: If the gscBsiGcUpdateValue() call returns the code BSI_NO_CARDSERVICE, then print
"gscBsiGcUpdateValue() is not supported.
Status: Test 18.4 Not Supported."

Case 3: If the gscBsiGcUpdateValue() call does not return the code BSI_BAD_AID or the code BSI_NO_CARDSERVICE, then:

Print
"gscBsiGcUpdateValue() called with a bad AID value returned an incorrect code.
Status: Test 18.4 Failed."

Test for Assertion 18.5

The function is tested using a bad AID length.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGcUpdateValue().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1805.

3. (Pre) There exists a target container on the connected card with the following properties:
   - the ACR for the gscBsiGcUpdateValue() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by AID value ==
     - _goodGSCAID1, of length _goodGSCAID1Len (GSC), or
     - _goodCACAID1, of length _goodCACAID1Len (CAC)
   - the container contains a data item for which
     - the tag is == _existingTag1805
     - the value is == _existingDvalue1805.

4. (Pre) Print "Testing of Assertion 18.5".

5. (Pre) Establish an authenticated session with the target container on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard1805
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - strctAuthenticator == strctAuthenticator1800
   - unAuthNb == 1.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 6.

   Case 2: If the gscBsiUtilAcquireContext() call does not return the code BSI_OK, then print
   "A session cannot be established. Assertion 18.5 of gscBsiGcUpdateValue() cannot be tested".
   End Test for Assertion 18.5.

6. Make a gscBsiGcUpdateValue() call to the SPS, using
   - hCard == hCard1805
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _badAIDLen
   - ucTag == _existingTag1805
   - uszValue == _newDvalue1805
   - unValueLen == _newDvalueLen.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_BAD_PARAM or the return code BSI_NO_CARDSERVICE.

2. No changes are made to the container structure of the connected card.

Perform this verification by issuing a call to gscBsiGcReadValue().

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiGcUpdateValue() call returns the code BSI_BAD_PARAM, then:

   Perform Test for Assertion 9.18.5.1 using
   - hCard == hCard1805.

   Allocate the unsigned char* uszValue1805 with length _newDvalueLen.

   Make a gscBsiGcReadValue() call to the SPS, using
   - hCard == hCard1805
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - ucTag == _existingTag1805
   - uszValue == uszValue1805
   - punValueLen == _newDvalueLen.

Case 1.1: If the gscBsiGcReadValue() call returns
- the code BSI_OK
- uszValue1805 == _existingDvalue1805

then:

Print
"gscBsiGcUpdateValue() called with a bad AID length has been verified because a subsequent call to gscBsiGcReadValue() found the original data item.
Status: Test 18.5 Passed."

Case 1.2: If the gscBsiReadValue() call returns
- the code BSI_OK
- uszValue1805 /= _existingDvalue1805

then:

Print
"gscBsiGcUpdateValue() called with a bad AID length has not been verified because a subsequent call to gscBsiGcReadValue() found that the data item had been updated or otherwise changed.
Status: Test 18.5 Failed."

Case 1.3: If the gscBsiGcReadValue() call returns any code other than
- BSI_OK

then:

Print
"gscBsiGcUpdateValue() called with a bad AID length has not been verified because a subsequent call to gscBsiGcReadValue() was ambiguous.
Status: Test 18.5 Undetermined."

Case 2: If the gscBsiGcUpdateValue() call returns the code BSI_NO_CARDSERVICE, then print
"gscBsiGcUpdateValue() is not supported.
Status: Test 18.5 Not Supported."

Case 3: If the gscBsiGcUpdateValue() call does not return the code BSI_BAD_PARAM or the code BSI_NO_CARDSERVICE, then:

Print
"gscBsiGcUpdateValue() called with a bad AID length returned an incorrect code.
Status: Test 18.5 Failed."

Test for Assertion 18.6

The function is tested using a bad data value length.

Instantiation Scenario:
1. (Pre) Construct the Starting State for the testing of gscBsiGcUpdateValue().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1806.

3. (Pre) There exists a target container on the connected card with the following properties:
   - the ACR for the gscBsiGcUpdateValue() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by AID value ==
     - _goodGSCAID1, of length _goodGSCAID1Len (GSC), or
     - _goodCACAID1, of length _goodCACAID1Len (CAC)
   - the container contains a data item for which
     - the tag is == _existingTag1806
     - the value is == _existingDvalue1806.

4. (Pre) Print "Testing of Assertion 18.6".

5. (Pre) Establish an authenticated session with the target container on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard1806
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - strctAuthenticator == strctAuthenticator1800
   - unAuthNb == 1.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 6.

   Case 2: If the gscBsiUtilAcquireContext() call does not return the code BSI_OK, then print
   "A session cannot be established. Assertion 18.6 of gscBsiGcUpdateValue() cannot be tested".
   End Test for Assertion 18.6.

6. Make a gscBsiGcUpdateValue() call to the SPS, using
   - hCard == hCard1806
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - ucTag == _existingTag1806
   - uszValue == _newDvalue1806
   - unValueLen == _badDvalueLen.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_BAD_PARAM or the return code BSI_NO_CARDSERVICE.

2. No changes are made to the container structure of the connected card.

Perform this verification by issuing a call to gscBsiGcReadValue().

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiGcUpdateValue() call returns the code BSI_BAD_PARAM, then:

   Perform Test for Assertion 9.18.6.1 using
   - hCard == hCard1806.

   Allocate the unsigned char* uszValue1806 with length _badDvalueLen.

   Make a gscBsiGcReadValue() call to the SPS, using
   - hCard == hCard1806
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - ucTag == _existingTag1806
   - uszValue == uszValue1806
   - punValueLen == _badDvalueLen.

   Case 1.1: If the gscBsiGcReadValue() call returns
   - the code BSI_OK
   - uszValue1806 == _existingDvalue1806

   then:

   Print
   "gscBsiGcUpdateValue() called with a bad data value length has been verified because a subsequent call to gscBsiGcReadValue() found the original data item.
   Status: Test 18.6 Passed."

   Case 1.2: If the gscBsiReadValue() call returns
   - the code BSI_OK
   - uszValue1806 /= _existingDvalue1806

   then:

   Print
   "gscBsiGcUpdateValue() called with a bad data value length has not been verified because a subsequent call to gscBsiGcReadValue() found that the data item had been updated or otherwise changed.
   Status: Test 18.6 Failed."

   Case 1.3: If the gscBsiGcReadValue() call returns any code other than
   - BSI_OK

   then:

   Print
   "gscBsiGcUpdateValue() called with a bad data value length has not been verified because a subsequent call to gscBsiGcReadValue() was ambiguous.
   Status: Test 18.6 Undetermined."

Case 2: If the gscBsiGcUpdateValue() call returns the code BSI_NO_CARDSERVICE, then print
"gscBsiGcUpdateValue() is not supported.
Status: Test 18.6 Not Supported."

Case 3: If the gscBsiGcUpdateValue() call does not return the code BSI_BAD_PARAM or the code BSI_NO_CARDSERVICE, then:

Print
"gscBsiGcUpdateValue() called with a bad data value length returned an incorrect code.
Status: Test 18.6 Failed."

Test for Assertion 18.7

The function is tested using a bad tag.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGcUpdateValue().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1807.

3. (Pre) There exists a target container on the connected card with the following properties:
   - the ACR for the gscBsiGcUpdateValue() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by AID value ==
     - _goodGSCAID1, of length _goodGSCAID1Len (GSC), or
     - _goodCACAID1, of length _goodCACAID1Len (CAC)
   - the container contains a data item for which
     - the tag is == _existingTag1807
     - the value is == _existingDvalue1807
   - the container does not contain a data item for which the tag == _newTag1807.

4. (Pre) Print "Testing of Assertion 18.7".

5. (Pre) Establish an authenticated session with the target container on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard1807
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - strctAuthenticator == strctAuthenticator1800
   - unAuthNb == 1.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 6.

   Case 2: If the gscBsiUtilAcquireContext() call does not return the code BSI_OK, then print
   "A session cannot be established. Assertion 18.7 of gscBsiGcUpdateValue() cannot be tested".
   End Test for Assertion 18.7.

6. Make a gscBsiGcUpdateValue() call to the SPS, using

- hCard == hCard1807
- uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
- unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
- ucTag == _newTag1807
- uszValue == _newDvalue1807
- unValueLen == _newDvalueLen.

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_BAD_TAG or the return code BSI_NO_CARDSERVICE.

2. No changes are made to the container structure of the connected card.

Perform this verification by issuing a call to gscBsiGcReadValue().

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiGcUpdateValue() call returns the code BSI_BAD_TAG, then:

   Perform Test for Assertion 9.18.7.1 using
   - hCard == hCard1807.

   Verify that a data item with the specified tag was not created:

   Allocate the unsigned char* uszValue1807 with length _newDvalueLen.

   Make a gscBsiGcReadValue() call to the SPS, using
   - hCard == hCard1807
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - ucTag == _newTag1807
   - uszValue == uszValue1807
   - punValueLen == _newDvalueLen.

   Case 1.1: If the gscBsiGcReadValue() call returns
   - the code BSI_OK

   then print:
   "gscBsiGcUpdateValue() called with a bad tag has not been verified because a subsequent call to gscBsiGcReadValue() found a data item with the specified tag.
   Status: Test 18.7 Failed."

   Case 1.2: If the gscBsiReadValue() call returns
   - the code BSI_BAD_TAG

then print:
   "gscBsiGcUpdateValue() called with a bad tag has been verified because a subsequent call to gscBsiGcReadValue() did not find a data item with the specified tag.
   Status: Test 18.7 Passed."

   Case 1.3: If the gscBsiGcReadValue() call returns any code other than

- BSI_OK or BSI_BAD_TAG

then print:
"gscBsiGcUpdateValue() called with a bad tag has not been verified because a subsequent call to gscBsiGcReadValue() was ambiguous.
Status: Test 18.7 Undetermined."

Case 2: If the gscBsiGcUpdateValue() call returns the code BSI_NO_CARDSERVICE, then print
"gscBsiGcUpdateValue() is not supported.
Status: Test 18.7 Not Supported."

Case 3: If the gscBsiGcUpdateValue() call does not return the code BSI_BAD_TAG or the code BSI_NO_CARDSERVICE, then:

Print
"gscBsiGcUpdateValue() called with a bad tag returned an incorrect code.
Status: Test 18.7 Failed."

Test for Assertion 18.8

The function is tested with a removed card.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGcUpdateValue().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1808.

3. (Pre) There exists a target container on the connected card with the following properties:
   - the ACR for the gscBsiGcUpdateValue() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by AID value ==
     - _goodGSCAID1, of length _goodGSCAID1Len (GSC), or
     - _goodCACAID1, of length _goodCACAID1Len (CAC)
   - the container contains a data item for which
     - the tag is == _existingTag1808
     - the value is == _existingDvalue1808.

4. (Pre) Print "Testing of Assertion 18.8".

5. (Pre) Establish an authenticated session with the target container on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard1808
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - strctAuthenticator == strctAuthenticator1800
   - unAuthNb == 1.

Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 6.

Case 2: If the gscBsiUtilAcquireContext() call does not return the code BSI_OK, then print
"A session cannot be established. Assertion 18.8 of gscBsiGcUpdateValue() cannot be tested".
End Test for Assertion 18.8.

6. (Pre) Remove the connected card from the reader.

7. Make a gscBsiGcUpdateValue() call to the SPS, using
   - hCard == hCard1808
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - ucTag == _existingTag1808
   - uszValue == _newDvalue1808
   - unValueLen == _newDvalueLen.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_CARD_REMOVED or the return code BSI_NO_CARDSERVICE.

2. No changes are made to the container structure of the connected card.

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiGcUpdateValue() call returns the code BSI_CARD_REMOVED, then:

   Print
   "gscBsiGcUpdateValue() called with the connected card removed has been verified.
   Status: Test 18.8 Passed."

   Case 2: If the gscBsiGcUpdateValue() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiGcUpdateValue() is not supported.
   Status: Test 18.8 Not Supported."

   Case 2: If the gscBsiGcUpdateValue() call does not return the code BSI_CARD_REMOVED or the code BSI_NO_CARDSERVICE, then:

   Print
   "gscBsiGcUpdateValue() called with the connected card removed returned an incorrect code.
   Status: Test 18.8 Failed."

Test for Assertion 18.9

The function is tested without fulfilling the applicable ACR.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGcUpdateValue().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1809.

3. (Pre) There exists a target container on the connected card with the following properties:
   - the ACR for the gscBsiGcUpdateValue() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by AID value ==
     - _goodGSCAID1, of length _goodGSCAID1Len (GSC), or
     - _goodCACAID1, of length _goodCACAID1Len (CAC)
   - the container contains a data item for which
     - the tag is == _existingTag1809
     - the value is == _existingDvalue1809.

4. (Pre) Print "Testing of Assertion 18.9".

5. (Pre) Ensure that there is no authenticated session with the target container:

   Make a gscBsiUtilReleaseContext() call to the SPS, using
   - hCard == hCard1809
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC).

6. Make a gscBsiGcUpdateValue() call to the SPS, using
   - hCard == hCard1809
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - ucTag == _existingTag1809
   - uszValue == _newDvalue1809
   - unValueLen == _newDvalueLen.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_ACCESS_DENIED or the return code BSI_NO_CARDSERVICE.

2. No changes are made to the container structure of the connected card.

Perform this verification by issuing a call to gscBsiGcReadValue().

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiGcUpdateValue() call returns the code BSI_ACCESS_DENIED, then:

   Perform Test for Assertion 9.18.9.1 using
   - hCard == hCard1809.

   Verify that that the specified data value in the target container was not changed:

Establish an authenticated session with the target container:

Make a gscBsiUtilAcquireContext() call to the SPS, using

- hCard == hCard1808
- AID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
- unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
- strctAuthenticator == strctAuthenticator1800
- unAuthNb == 1.

Case 1.1: If the gscBsiUtilAcquireContext call returns the code BSI_OK, then

Allocate the unsigned char* uszValue1809 with length _newDvalueLen.

Make a gscBsiGcReadValue() call to the SPS, using

- hCard == hCard1809
- uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
- unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
- ucTag == _existingTag1809
- uszValue == uszValue1809
- punValueLen == _newDvalueLen.

Case 1.1.1: If the gscBsiGcReadValue() call returns

- the code BSI_OK
- uszValue1809 == _existingDvalue1809

then print

"gscBsiGcUpdateValue() called without fulfilling the applicable ACR has been verified because a subsequent call to gscBsiGcReadValue() found the original data item.
Status: Test 18.9 Passed."

Case 1.1.2: If the gscBsiReadValue() call returns

- the code BSI_OK
- uszValue1809 /= _existingDvalue1809

then print

"gscBsiGcUpdateValue() called without fulfilling the applicable ACR has not been verified because a subsequent call to gscBsiGcReadValue() found that the data item had been updated or otherwise changed.
Status: Test 18.9 Failed."

Case 1.1.3: If the gscBsiGcReadValue() call does not return the code

- BSI_OK

then print

"gscBsiGcUpdateValue() called without fulfilling the applicable ACR has not been verified because a subsequent call to gscBsiGcReadValue() was ambiguous.
Status: Test 18.9 Undetermined."

Case 1.2: If the gscBsiUtilAcquireContext call does not return the code BSI_OK, then print

"A session with the second container could not be established. gscBsiUtilDataUpdate() cannot be verified. **Status**: Test 18.9 Undetermined."

Case 2: If the gscBsiGcUpdateValue() call returns the code BSI_NO_CARDSERVICE, then print
"gscBsiGcUpdateValue() is not supported.
Status: Test 18.9 Not Supported."

Case 3: If the gscBsiGcUpdateValue() call does not return the code BSI_ACCESS_DENIED or the code BSI_NO_CARDSERVICE, then print
"gscBsiGcUpdateValue() called without fulfilling the applicable ACR returned an incorrect code.
Status: Test 18.9 Failed."

Test for Assertion 18.10

The function is tested using a too-large data value.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGcUpdateValue().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1810.

3. (Pre) There exists a target container on the connected card with the following properties:
   - the ACR for the gscBsiGcUpdateValue() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by AID value ==
     - _goodGSCAID2, of length _goodGSCAID2Len (GSC), or
     - _goodCACAID2, of length _goodCACAID2Len (CAC)
   - the container contains one data item, which comprises the entire available space of the container
     - the tag for this data item is _existingTagFull
     - the value of this data item is _existingDvalueFull.

4. (Pre) Print "Testing of Assertion 18.10".

5. (Pre) Establish an authenticated session with the target container on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard1810
   - uszAID == _goodGSCAID2 (GSC) or _goodCACAID2 (CAC)
   - unAIDLen == _goodGSCAID2Len (GSC) or _goodCACAID2Len (CAC)
   - strctAuthenticator == strctAuthenticator1800
   - unAuthNb == 1.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 6.

Case 2: If the gscBsiUtilAcquireContext() call does not return the code BSI_OK, then print

"A session cannot be established. Assertion 18.10 of gscBsiGcUpdateValue() cannot be tested".

End Test for Assertion 18.10.

6. Make a gscBsiGcUpdateValue() call to the SPS, using
   - hCard == hCard1810
   - uszAID == _goodGSCAID2 (GSC) or _goodCACAID2 (CAC)
   - unAIDLen == _goodGSCAID2Len (GSC) or _goodCACAID2Len (CAC)
   - ucTag == _existingTagFull
   - uszValue == _tooBigDvalue
   - unValueLen == _tooBigDvalueLen.

Verification Goal:

To verify the Expected Results:

1. The call returns
   - the return code BSI_NO_MORE_SPACE or the return code BSI_NO_CARDSERVICE.

2. No changes are made to the container structure of the connected card.

Perform this verification by issuing a call to gscBsiGcReadValue().

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiGcUpdateValue() call returns the code BSI_NO_MORE_SPACE, then:

   Perform Test for Assertion 9.18.10.1 using
   - hCard == hCard1810.

   Allocate the unsigned char* uszValue1810 with length _existingDvalueFullLen.

   Make a gscBsiGcReadValue() call to the SPS, using
   - hCard == hCard1810
   - uszAID == _goodGSCAID2 (GSC) or _goodCACAID2 (CAC)
   - unAIDLen == _goodGSCAID2Len (GSC) or _goodCACAID2Len (CAC)
   - ucTag == _existingTagFull
   - uszValue == uszValue1810
   - punValueLen == _existingDvalueFullLen.

   Case 1.1: If the gscBsiGcReadValue() call returns
   - the code BSI_OK
   - uszValue1810 == _existingDvalueFull

   then:

   Print

   "gscBsiGcUpdateValue() called using a too-large data value has been verified because a subsequent call to gscBsiGcReadValue() found the original data item. **Status**: Test 18.10 Passed."

   Case 1.2: If the gscBsiReadValue() call returns
   - the code BSI_OK

- uszValue1810 /= _existingDvalueFull

then:

Print
"gscBsiGcUpdateValue() called using a too-large data value has not been verified because a subsequent call to gscBsiGcReadValue() found that the data item had been updated or otherwise changed.
Status: Test 18.10 Failed."

Case 1.3: If the gscBsiGcReadValue() call returns any code other than

- BSI_OK

then:

Print
"gscBsiGcUpdateValue()called using a too-large data value has not been verified because a subsequent call to gscBsiGcReadValue() was ambiguous.
Status: Test 18.10 Undetermined."

Case 2: If the gscBsiGcUpdateValue() call returns the code BSI_NO_CARDSERVICE, then print
"gscBsiGcUpdateValue() is not supported.
Status: Test 18.10 Not Supported."

Case 3: If the gscBsiGcUpdateValue() call does not return the code BSI_NO_MORE_SPACE or the code BSI_NO_CARDSERVICE, then:

Print
"gscBsiGcUpdateValue() called using a too-large data value returned an incorrect code.
Status: Test 18.10 Failed."

19. gscBsiGetChallenge()

Test for Assertion 19.1

The function is tested using valid parameters (Discovery Method 1).

Instantiation Scenario:

1. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1901.

2. (Pre) There exists a target container on the connected card. The container is represented by AID value ==
   - _goodGSCAID1, of length _goodGSCAID1Len (GSC), or
   - _goodCACAID1, of length _goodCACAID1Len (CAC).

3. (Pre) Allocate the unsigned long* punChallengeLen1901 == 0.

4. (Pre) Print "Testing of Assertion 19.1".

5. Make a gscBsiGetChallenge() call to the SPS, using
   - hCard == hCard1901
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - uszChallenge == NULL
   - punChallengeLen == punChallengeLen1901.

   Case 1: If the gscBsiGetChallenge() call returns the code BSI_OK, then print
   "gscBsiGetChallenge() (Discovery Method 1, Discovery Mode) succeeded."
   Continue with 6.

   Case 2: If the gscBsiGetChallenge() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiGetChallenge() is not supported.
   Status: Test 19.1 Not Supported."
   End Test for Assertion 19.1.

   Case 3: If the gscBsiGetChallenge() call does not return the code BSI_OK or the code BSI_NO_CARDSERVICE, then print
   "gscBsiGetChallenge() (Discovery Method 1, Discovery Mode) returned an incorrect code.
   Status: Test 19.1 Failed."
   End Test for Assertion 19.1.

6. Allocate the unsigned string uszChallenge1901 with length == punChallengeLen1901*.

7. Make a gscBsiGetChallenge() call to the SPS, using
   - hCard == hCard1901
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - uszChallenge == uszChallenge1901
   - punChallengeLen == punChallengeLen1901.

Verification Goal:

To verify the Expected Results:

1. The call returns
   - the return code BSI_OK or the return code BSI_NO_CARDSERVICE

2. If the return code is BSI_OK, then uszChallenge1901 == a string containing the random challenge returned from the connected card.

Perform this verification by inspection.

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiGetChallenge() call returns the code BSI_OK, then manually inspect the string uszChallenge1901.

   Case 1.1: If uszChallenge1901 contains a valid random challenge, then print
   "gscBsiGetChallenge() (Discovery Method 1, Final Mode) called with valid parameters has been verified by inspection.
   Status: Test 19.1 Passed."

   Case 1.2: If uszChallenge1901 does not contain a valid random challenge, then print
   "gscBsiGetChallenge() (Discovery Method 1, Final Mode) called with valid parameters has not been verified by inspection.
   Status: Test 19.1 Failed."

   Case 2: If the gscBsiGetChallenge() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiGetChallenge() is not supported.
   Status: Test 19.1 Not Supported."

   Case 3: If the gscBsiGetChallenge() call does not return the code BSI_OK or the code BSI_NO_CARDSERVICE, then print
   "gscBsiGetChallenge() (Discovery Method 1, Final Mode) called with valid parameters returned an incorrect code.
   Status: Test 19.1 Failed."

Test for Assertion 19.2

The function is tested using valid parameters (Discovery Method 2).

Instantiation Scenario:

1. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1902.

2. (Pre) There exists a target container on the connected card. The container is represented by AID value ==
   - _goodGSCAID1, of length _goodGSCAID1Len (GSC), or
   - _goodCACAID1, of length _goodCACAID1Len (CAC).

3. (Pre) Allocate the string uszChallenge1902 with length == 0.

4. (Pre) Allocate the unsigned long* punChallengeLen1902 == 0.

5. (Pre) Print "Testing of Assertion 19.2".

6. Make a gscBsiGetChallenge() call to the SPS, using

- hCard == hCard1902
- uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
- unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
- uszChallenge == uszChallenge1902
- punChallengeLen == punChallengeLen1902.

Case 1: If the gscBsiGetChallenge() call returns the code BSI_INSUFFICIENT_BUFFER, then print
"gscBsiGetChallenge() (Discovery Method 2, Discovery Mode) correctly returned the code BSI_INSUFFICIENT_BUFFER"
Continue with 7.

Case 2: If the gscBsiGetChallenge() call returns the code BSI_NO_CARDSERVICE, then print
"gscBsiGetChallenge() is not supported.
Status: Test 19.2 Not Supported."
End Test for Assertion 19.2.

Case 3: If the gscBsiGetChallenge() call does not return the code BSI_INSUFFICIENT_BUFFER or the code BSI_NO_CARDSERVICE, then print
"gscBsiGetChallenge() (Discovery Method 2, Discovery Mode) returned an incorrect code.
Status: Test 19.2 Failed."
End Test for Assertion 19.2.

7. Re-allocate the string uszChallenge1902 with length == punChallengeLen1902*.

8. Make a gscBsiGetChallenge() call to the SPS, using
   - hCard == hCard1902
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - uszChallenge == uszChallenge1902
   - punChallengeLen == punChallengeLen1902.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_OK or the return code BSI_NO_CARDSERVICE.

2. If the return code is BSI_OK, then uszChallenge1902 == a string containing the random challenge returned from the connected card.

Perform this verification by inspection.

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiGetChallenge() call returns the code BSI_OK, then manually inspect the string uszChallenge1902.

   Case 1.1: If uszChallenge1902 contains a valid random challenge, then print
   "gscBsiGetChallenge() (Discovery Method 2, Final Mode) called with valid parameters has been verified by inspection.
   Status: Test 19.2 Passed."

Case 1.2: If uszChallenge1902 does not contain a valid random challenge, then print
"gscBsiGetChallenge() (Discovery Method 2, Final Mode) called with valid parameters has not been verified by inspection.
Status: Test 19.2 Failed."

Case 2: If the gscBsiGetChallenge() call returns the code BSI_NO_CARDSERVICE, then print
"gscBsiGetChallenge() is not supported.
Status: Test 19.2 Not Supported."

Case 3: If the gscBsiGetChallenge() call does not return the code BSI_OK or the code BSI_NO_CARDSERVICE, then print
"gscBsiGetChallenge() (Discovery Method 2, Final Mode) called with valid parameters returned an incorrect code.
Status: Test 19.2 Failed."

Test for Assertion 19.3

The function is tested using a bad handle (Discovery Method 1, Discovery Mode).

Instantiation Scenario:

1. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1903.

2. (Pre) There exists a target container on the connected card. The container is represented by AID value ==
   - _goodGSCAID1, of length _goodGSCAID1Len (GSC), or
   - _goodCACAID1, of length _goodCACAID1Len (CAC).

3. (Pre) Allocate the unsigned long* punChallengeLen1903 == 0.

4. (Pre) Print "Testing of Assertion 19.3".

5. Make a gscBsiGetChallenge() call to the SPS, using
   - hCard /= hCard1903
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - uszChallenge == NULL
   - punChallengeLen == punChallengeLen1903.

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_BAD_HANDLE or the return code BSI_NO_CARDSERVICE.

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiGetChallenge() call returns the code BSI_BAD_HANDLE, then:

   Perform Test for Assertion 9.19.3.1 using
   - hCard == hCard1903.

   Print

"gscBsiGetChallenge() (Discovery Method 1, Discovery Mode) called with a bad handle has been verified.
Status: Test 19.3 Passed."

Case 2: If the gscBsiGetChallenge() call returns the code BSI_NO_CARDSERVICE, then print
"gscBsiGetChallenge() is not supported.
Status: Test 19.3 Not Supported."

Case 3: If the gscBsiGetChallenge() call does not return the code BSI_BAD_HANDLE or the code BSI_NO_CARDSERVICE, then print
"gscBsiGetChallenge() (Discovery Method 1, Discovery Mode) called with a bad handle returned an incorrect code.
Status: Test 19.3 Failed."

Test for Assertion 19.4

The function is tested using a bad handle (Discovery Method 1, Final Mode).

Instantiation Scenario:

1. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1904.

2. (Pre) There exists a target container on the connected card. The container is represented by AID value ==
   - _goodGSCAID1, of length _goodGSCAID1Len (GSC), or
   - _goodCACAID1, of length _goodCACAID1Len (CAC).

3. (Pre) Allocate the unsigned long* punChallengeLen1904 == 0.

4. (Pre) Print "Testing of Assertion 19.4".

5. Make a gscBsiGetChallenge() call to the SPS, using
   - hCard == hCard1904
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - uszChallenge == NULL
   - punChallengeLen == punChallengeLen1904.

   Case 1: If the gscBsiGetChallenge() call returns the code BSI_OK, then print
   "gscBsiGetChallenge() (Discovery Method 1, Discovery Mode) succeeded."
   Continue with 6.

   Case 2: If the gscBsiGetChallenge() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiGetChallenge() is not supported.
   Status: Test 19.4 Not Supported."
   End Test for Assertion 19.4.

   Case 3: If the gscBsiGetChallenge() call does not return the code BSI_OK or the code BSI_NO_CARDSERVICE, then print
   "gscBsiGetChallenge() (Discovery Method 1, Discovery Mode) did not return the code BSI_OK.

Status: Test 19.4 Failed."
End Test for Assertion 19.4.

6. Allocate the string uszChallenge1904 with length == punChallengeLen1904*.

7. Make a gscBsiGetChallenge() call to the SPS, using
   - hCard /= hCard1904
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - uszChallenge == uszChallenge1904
   - punChallengeLen == punChallengeLen1904.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_BAD_HANDLE or the return code BSI_NO_CARDSERVICE.

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiGetChallenge() call returns the code BSI_BAD_HANDLE, then:

   Perform Test for Assertion 9.19.4.1

   Print
   "gscBsiGetChallenge() (Discovery Method 1, Final Mode) called with a bad handle has been verified.
   Status: Test 19.4 Passed."

   Case 2: If the gscBsiGetChallenge() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiGetChallenge() is not supported.
   Status: Test 19.4 Not Supported."

   Case 3: If the gscBsiGetChallenge() call does not return the code BSI_BAD_HANDLE or the code BSI_NO_CARDSERVICE, then print
   "gscBsiGetChallenge() (Discovery Method 1, Final Mode) called with a bad handle returned an incorrect code.
   Status: Test 19.4 Failed."

Test for Assertion 19.5

The function is tested with another application having established a transaction lock (Discovery Method 1, Discovery Mode).

Until we encounter implementations that allow multiple simultaneous applications, I don't think we need to worry about this assertion.

Test for Assertion 19.6

The function is tested using a bad AID value (Discovery Method 1, Discovery Mode).

Instantiation Scenario:

1. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1906.

2. (Pre) There exists a target container on the connected card. The container is represented by AID value ==
   - _goodGSCAID1, of length _goodGSCAID1Len (GSC), or
   - _goodCACAID1, of length _goodCACAID1Len (CAC).

3. (Pre) There does not exist a container on the connected card with AID value ==
   - _badGSCAID, of length _badGSCAIDLen (GSC), or
   - _badCACAID, of length _badCACAIDLen (CAC).

4. (Pre) Allocate the unsigned long* punChallengeLen1906 == 0.

5. (Pre) Print "Testing of Assertion 19.6".

6. Make a gscBsiGetChallenge() call to the SPS, using
   - hCard == hCard1906
   - uszAID == _badGSCAID (GSC) or _badCACAID (CAC)
   - unAIDLen == _badGSCAIDLen (GSC) or _badCACAIDLen (CAC)
   - uszChallenge == NULL
   - punChallengeLen == punChallengeLen1906.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_BAD_AID or the return code BSI_NO_CARDSERVICE.

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiGetChallenge() call returns the code BSI_BAD_AID, then:

   Perform Test for Assertion 9.19.6.1 using
   - hCard == hCard1906.

   Print
   "gscBsiGetChallenge() (Discovery Method 1, Discovery Mode) called with a bad AID value has been verified.
   Status: Test 19.6 Passed."

   Case 2: If the gscBsiGetChallenge() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiGetChallenge() is not supported.
   Status: Test 19.6 Not Supported."

   Case 3: If the gscBsiGetChallenge() call does not return the code BSI_BAD_AID or the code BSI_NO_CARDSERVICE, then print
   "gscBsiGetChallenge() (Discovery Method 1, Discovery Mode) called with a bad AID value returned an incorrect code.
   Status: Test 19.6 Failed."

Test for Assertion 19.7

The function is tested using a bad AID length (Discovery Method 1, Discovery Mode).

Instantiation Scenario:

1. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1907.

2. (Pre) There exists a target container on the connected card. The container is represented by AID value ==
   - _goodGSCAID1, of length _goodGSCAID1Len (GSC), or
   - _goodCACAID1, of length _goodCACAID1Len (CAC).

3. (Pre) Allocate the unsigned long* punChallengeLen1907 == 0.

4. (Pre) Print "Testing of Assertion 19.7".

5. Make a gscBsiGetChallenge() call to the SPS, using
   - hCard == hCard1907
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _badAIDLen
   - uszChallenge == NULL
   - punChallengeLen == punChallengeLen1907.

Verification Goal:

To verify the Expected Results:

1. The call returns
   - the return code BSI_BAD_PARAM or the return code BSI_NO_CARDSERVICE.

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiGetChallenge() call returns the code BSI_BAD_PARAM, then:

   Perform Test for Assertion 9.19.7.1 using
   - hCard == hCard1907.

   Print
   "gscBsiGetChallenge() (Discovery Method 1, Discovery Mode) called with a bad AID length has been verified.
   Status: Test 19.7 Passed."

   Case 2: If the gscBsiGetChallenge() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiGetChallenge() is not supported.
   Status: Test 19.7 Not Supported."

   Case 3: If the gscBsiGetChallenge() call does not return the code BSI_BAD_PARAM or the code BSI_NO_CARDSERVICE, then print
   "gscBsiGetChallenge() (Discovery Method 1, Discovery Mode) called with a bad AID length returned an incorrect code.
   Status: Test 19.7 Failed."

Test for Assertion 19.8

The function is tested with a removed card (Discovery Method 1, Discovery Mode).

Instantiation Scenario:

1. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard1908.

8. (Pre) There exists a target container on the connected card. The container is represented by AID value ==
   - _goodGSCAID1, of length _goodGSCAID1Len (GSC), or
   - _goodCACAID1, of length _goodCACAID1Len (CAC).

2. (Pre) Allocate the unsigned long* punChallengeLen1908 == 0.

3. (Pre) Print "Testing of Assertion 19.8".

4. Remove the connected card from the reader,

5. Make a gscBsiGetChallenge() call to the SPS, using
   - hCard == hCard1908
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - uszChallenge == NULL
   - punChallengeLen == punChallengeLen1908.

Verification Goal:

To verify the Expected Results:

1. The call returns
   - the return code BSI_CARD_REMOVED or the return code BSI_NO_CARDSERVICE.

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiGetChallenge() call returns the code BSI_CARD_REMOVED, then:

   Print
   "gscBsiGetChallenge() (Discovery Method 1, Discovery Mode) called with the connected card removed has been verified.
   Status: Test 19.8 Passed."

   Case 2: If the gscBsiGetChallenge() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiGetChallenge() is not supported.
   Status: Test 19.8 Not Supported."

   Case 2: If the gscBsiGetChallenge() call does not return the code BSI_CARD_REMOVED or the code BSI_NO_CARDSERVICE, then print
   "gscBsiGetChallenge() (Discovery Method 1, Discovery Mode) called with the connected card removed returned an incorrect code.
   Status: Test 19.8 Failed."

20. gscBsiSkiInternalAuthenticate()

Starting State for Each Test:

1. There exists a BSIAuthenticator array strctAuthenticator2000 with one element, the structure BSIAuthenticator2000. This structure has fields
   - unAccessMethodType2000 == BSI_AM_PIN
   - unkeyIDOrReference2000 == _keyIDOrReference1
   - uszAuthValue2000 == _goodAuthValue1
   - unAuthValueLen2000 == _goodAuthValue1Len.

Test for Assertion 20.1

The function is tested using valid parameters (Discovery Method 1).

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiSkiInternalAuthenticate().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard2001.

3. (Pre) There exists a target SKI provider module on the connected card with the following properties:
   - the ACR for the gscBsiSkiInternalAuthenticate() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by AID value ==
     - _goodGSCAID2, of length _goodGSCAID2Len (GSC), or
     - _goodCACAID2, of length _goodCACAID2Len (CAC).

4. (Pre) Print "Testing of Assertion 20.1".

5. (Pre) Establish an authenticated session with the target SKI provider module on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard2001
   - uszAID == _goodGSCAID2 (GSC) or _goodCACAID2 (CAC)
   - unAIDLen == _goodGSCAID2Len (GSC) or _goodCACAID2Len (CAC)
   - strctAuthenticator == strctAuthenticator2000
   - unAuthNb == 1.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 6.

   Case 2: If the gscBsiUtilAcquireContext() call does not return the code BSI_OK, then print
   "A session cannot be established. Assertion 20.1 of gscBsiSkiInternalAuthenticate() cannot be tested".
   End Test for Assertion 20.1.

6. (Pre) Allocate the unsigned long* punCryptogramLen2001 == 0.

7. Make a gscBsiSkiInternalAuthenticate() call to the SPS, using
   - hCard == hCard2001
   - uszAID == _goodGSCAID2 (GSC) or _goodCACAID2 (CAC)
   - unAIDLen == _goodGSCAID2Len (GSC) or _goodCACAID2Len (CAC)
   - ucAlgoID == _goodAlgoID1
   - uszChallenge == _goodChallenge
   - unChallengeLen == _goodChallengeLen
   - uszCryptogram == NULL
   - punCryptogramLen == punCryptogramLen2001.

   Case 1: If the gscBsiSkiInternalAuthenticate() call returns the code BSI_OK, then print
   "gscBsiSkiInternalAuthenticate() (Discovery Method 1, Discovery Mode) succeeded."
   Continue with 8.

   Case 2: If the gscBsiSkiInternalAuthenticate() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiSkiInternalAuthenticate() is not supported. **Status**: Test 20.1 Not Supported."
   End Test for Assertion 20.1.

   Case 3: If the gscBsiSkiInternalAuthenticate() call does not return the code BSI_OK or the code BSI_NO_CARDSERVICE, then print
   "gscBsiSkiInternalAuthenticate() (Discovery Method 1, Discovery Mode) did not return the code BSI_OK. **Status**: Test 20.1 Failed."
   End Test for Assertion 20.1.

8. Allocate the string uszCryptogram2001 with length == punCryptogramLen2001*.

9. Make a gscBsiSkiInternalAuthenticate() call to the SPS, using
   - hCard == hCard2001
   - uszAID == _goodGSCAID2 (GSC) or _goodCACAID2 (CAC)
   - unAIDLen == _goodGSCAID2Len (GSC) or _goodCACAID2Len (CAC)
   - ucAlgoID == _goodAlgoID1
   - uszChallenge == _goodChallenge
   - unChallengeLen == _goodChallengeLen
   - uszCryptogram == uszCryptogram2001
   - punCryptogramLen == punCryptogramLen2001.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_OK or the return code BSI_NO_CARDSERVICE.

2. If the return code is BSI_OK, than uszCryptogram2001 == a string containing the symmetric key cryptogram returned from the connected card.

Perform this verification by inspection.

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiSkiInternalAuthenticate() call returns the code BSI_OK, then manually inspect uszCryptogram2001:

Case 1.1: If uszCryptogram2001 contains a valid symmetric key cryptogram, then print
"gscBsiSkiInternalAuthenticate() (Discovery Method 1, Final Mode) called with valid parameters has been verified.
Status: Test 20.1 Passed."

Case 1.2: If uszCryptogram2001 does not contain a valid symmetric key cryptogram, then print
"gscBsiSkiInternalAuthenticate() (Discovery Method 1, Final Mode) called with valid parameters has not been verified.
Status: Test 20.1 Failed."

Case 2: If the gscBsiSkiInternalAuthenticate() call returns the code BSI_NO_CARDSERVICE, then print
"gscBsiSkiInternalAuthenticate() is not supported.
Status: Test 20.1 Not Supported."

Case 3: If the gscBsiSkiInternalAuthenticate() call does not return the code BSI_OK or the code BSI_NO_CARDSERVICE, then print
"gscBsiSkiInternalAuthenticate() (Discovery Method 1, Final Mode) called with valid parameters returned an incorrect code.
Status: Test 20.1 Failed."

Test for Assertion 20.2

The function is tested using valid parameters (Discovery Method 2).

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiSkiInternalAuthenticate().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard2002.

3. (Pre) There exists a target SKI provider module on the connected card with the following properties:
   - the ACR for the gscBsiSkiInternalAuthenticate() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by AID value ==
     - _goodGSCAID2, of length _goodGSCAID2Len (GSC), or
     - _goodCACAID2, of length _goodCACAID2Len (CAC).

4. (Pre) Print "Testing of Assertion 20.2".

5. (Pre) Establish an authenticated session with the target SKI provider module on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard2002
   - uszAID == _goodGSCAID2 (GSC) or _goodCACAID2 (CAC)
   - unAIDLen == _goodGSCAID2Len (GSC) or _goodCACAID2Len (CAC)
   - strctAuthenticator == strctAuthenticator2000
   - unAuthNb == 1.

Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 6.

Case 2: If the gscBsiUtilAcquireContext() call does not return the code BSI_OK, then print
"A session cannot be established. Assertion 20.2 of gscBsiSkiInternalAuthenticate() cannot be tested".
End Test for Assertion 20.2.

6. (Pre) Allocate the string uszCryptogram2002 with length == 0.

7. (Pre) Allocate the unsigned long* punCryptogramLen2002 == 0.

8. Make a gscBsiSkiInternalAuthenticate() call to the SPS, using
   - hCard == hCard2002
   - uszAID == _goodGSCAID2 (GSC) or _goodCACAID2 (CAC)
   - unAIDLen == _goodGSCAID2Len (GSC) or _goodCACAID2Len (CAC)
   - ucAlgoID == _goodAlgoID1
   - uszChallenge == _goodChallenge
   - unChallengeLen == _goodChallengeLen
   - uszCryptogram == uszCryptogram2002
   - punCryptogramLen == punCryptogramLen2002.

   Case 1: If the gscBsiSkiInternalAuthenticate() call returns the code BSI_INSUFFICIENT_BUFFER, then print
   "gscBsiSkiInternalAuthenticate() (Discovery Method 2, Discovery Mode) succeeded."
   Continue with 9.

   Case 2: If the gscBsiSkiInternalAuthenticate() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiSkiInternalAuthenticate() is not supported.
   Status: Test 20.2 Not Supported."
   End Test for Assertion 20.2

   Case 3: If the gscBsiSkiInternalAuthenticate() call does not return the code BSI_INSUFFICIENT_BUFFER or the code BSI_NO_CARDSERVICE, then print
   "gscBsiSkiInternalAuthenticate() (Discovery Method 2, Discovery Mode) did not return the code BSI_INSUFFICIENT_BUFFER.
   Status: Test 20.2 Failed."
   End Test for Assertion 20.2.

9. Re-allocate the string uszCryptogram2002 with length == punCryptogramLen2002*.

10. Make a gscBsiSkiInternalAuthenticate() call to the SPS, using
    - hCard == hCard2002
    - uszAID == _goodGSCAID2 (GSC) or _goodCACAID2 (CAC)
    - unAIDLen == _goodGSCAID2Len (GSC) or _goodCACAID2Len (CAC)
    - ucAlgoID == _goodAlgoID1
    - uszChallenge == _goodChallenge
    - unChallengeLen == _goodChallengeLen
    - uszCryptogram == uszCryptogram2002
    - punCryptogramLen == punCryptogramLen2002.

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_OK or the return code BSI_NO_CARDSERVICE.

2. If the return code is BSI_OK, then uszCryptogram2002 == a string containing the symmetric key cryptogram returned from the connected card.

Perform this verification by inspection.

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiSkiInternalAuthenticate() call returns the code BSI_OK, then manually inspect uszCryptogram2002:

   Case 1.1: If uszCryptogram2002 contains a symmetric key cryptogram, then print
   "gscBsiSkiInternalAuthenticate() (Discovery Method 2, Final Mode) called with valid parameters has been verified.
   Status: Test 20.2 Passed."

   Case 1.2: If uszCryptogram2002 does not contain a symmetric key cryptogram, then print
   "gscBsiSkiInternalAuthenticate() (Discovery Method 2, Final Mode) called with valid parameters has not been verified.
   Status: Test 20.2 Failed."

   Case 2: If the gscBsiSkiInternalAuthenticate() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiSkiInternalAuthenticate() is not supported.
   Status: Test 20.2 Not Supported."

   Case 3: If the gscBsiSkiInternalAuthenticate() call does not return the code BSI_OK or the code BSI_NO_CARDSERVICE, then print
   "gscBsiSkiInternalAuthenticate() (Discovery Method 2, Final Mode) called with valid parameters returned an incorrect code.
   Status: Test 20.2 Failed."

Test for Assertion 20.3

The function is tested using a bad handle (Discovery Method 1, Discovery Mode).

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiSkiInternalAuthenticate().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard2003.

3. (Pre) There exists a target SKI provider module on the connected card with the following properties:
   - the ACR for the gscBsiSkiInternalAuthenticate() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN

- the container is represented by AID value ==
  - _goodGSCAID2, of length _goodGSCAID2Len (GSC), or
  - _goodCACAID2, of length _goodCACAID2Len (CAC).

4. (Pre) Print "Testing of Assertion 20.3".

5. (Pre) Establish an authenticated session with the target SKI provider module on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard2003
   - uszAID == _goodGSCAID2 (GSC) or _goodCACAID2 (CAC)
   - unAIDLen == _goodGSCAID2Len (GSC) or _goodCACAID2Len (CAC)
   - strctAuthenticator == strctAuthenticator2000
   - unAuthNb == 1.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 6.

   Case 2: If e gscBsiUtilAcquireContext() call does not return the code BSI_OK, then print
   "A session cannot be established. Assertion 20.3 of gscBsiSkiInternalAuthenticate() cannot be tested".
   End Test for Assertion 20.3.

6. (Pre) Allocate the unsigned long* punCryptogramLen2003 == 0.

7. Make a gscBsiSkiInternalAuthenticate() call to the SPS, using
   - hCard /= hCard2003
   - uszAID == _goodGSCAID2 (GSC) or _goodCACAID2 (CAC)
   - unAIDLen == _goodGSCAID2Len (GSC) or _goodCACAID2Len (CAC)
   - ucAlgoID == _goodAlgoID1
   - uszChallenge == _goodChallenge
   - unChallengeLen == _goodChallengeLen
   - uszCryptogram == NULL
   - punCryptogramLen == punCryptogramLen2003.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_BAD_HANDLE or the return code BSI_NO_CARDSERVICE.

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiSkiInternalAuthenticate() call returns the code BSI_BAD_HANDLE, then:

   Perform test for Assertion 9.20.3.1 using
   - hCard == hCard2003.

   Print
   "gscBsiSkiInternalAuthenticate() (Discovery Method 1, Discovery Mode) called with a bad handle has been verified. **Status**: Test 20.3 Passed."

Case 2: If the gscBsiSkiInternalAuthenticate() call returns the code BSI_NO_CARDSERVICE, then print
"gscBsiSkiInternalAuthenticate() is not supported.
Status: Test 20.3 Not Supported."

Case 3: If the gscBsiSkiInternalAuthenticate() call does not return the code BSI_BAD_HANDLE or the code BSI_NO_CARDSERVICE, then print
"gscBsiSkiInternalAuthenticate() (Discovery Method 1, Discovery Mode) called with a bad handle returned an incorrect code.
Status: Test 20.3 Failed."

Test for Assertion 20.4

The function is tested using a bad handle (Discovery Method 1, Final Mode).

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiSkiInternalAuthenticate().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard2004.

3. (Pre) There exists a target SKI provider module on the connected card with the following properties:
   - the ACR for the gscBsiSkiInternalAuthenticate() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by AID value ==
     - _goodGSCAID2, of length _goodGSCAID2Len (GSC), or
     - _goodCACAID2, of length _goodCACAID2Len (CAC).

4. (Pre) Print "Testing of Assertion 20.4".

5. (Pre) Establish an authenticated session with the target SKI provider module on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard2004
   - uszAID == _goodGSCAID2 (GSC) or _goodCACAID2 (CAC)
   - unAIDLen == _goodGSCAID2Len (GSC) or _goodCACAID2Len (CAC)
   - strctAuthenticator == strctAuthenticator2000
   - unAuthNb == 1.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 6.

   Case 2: If the gscBsiUtilAcquireContext() call does not return the code BSI_OK, then print
   "A session cannot be established. Assertion 20.4 of gscBsiSkiInternalAuthenticate() cannot be tested".
   End Test for Assertion 20.4.

6. (Pre) Allocate the unsigned long* punCryptogramLen2004 == 0.

7. Make a gscBsiSkiInternalAuthenticate() call to the SPS, using
   - hCard == hCard2004
   - uszAID == _goodGSCAID2 (GSC) or _goodCACAID2 (CAC)
   - unAIDLen == _goodGSCAID2Len (GSC) or _goodCACAID2Len (CAC)
   - ucAlgoID == _goodAlgoID1
   - uszChallenge == _goodChallenge
   - unChallengeLen == _goodChallengeLen
   - uszCryptogram == NULL
   - punCryptogramLen == punCryptogramLen2004.

   Case 1: If the gscBsiSkiInternalAuthenticate() call returns the code BSI_INSUFFICIENT_BUFFER, then print
   "gscBsiSkiInternalAuthenticate() (Discovery Method 2, Discovery Mode) succeeded."
   Continue with 8.

   Case 2: If the gscBsiSkiInternalAuthenticate() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiSkiInternalAuthenticate() is not supported.
   Status: Test 20.4 Not Supported."
   End Test for Assertion 20.4.

   Case 3: If the gscBsiSkiInternalAuthenticate() call does not return the code BSI_INSUFFICIENT_BUFFER or the code BSI_NO_CARDSERVICE, then print
   "gscBsiSkiInternalAuthenticate() (Discovery Method 2, Discovery Mode) did not return the code BSI_INSUFFICIENT_BUFFER.
   Status: Test 20.4 Failed."
   End Test for Assertion 20.4.

8. Allocate the stringuszCryptogram2004 with length == punCryptogram2004*.

9. Make a gscBsiSkiInternalAuthenticate() call to the SPS, using
   - hCard /= hCard2004
   - uszAID == _goodGSCAID2 (GSC) or _goodCACAID2 (CAC)
   - unAIDLen == _goodGSCAID2Len (GSC) or _goodCACAID2Len (CAC)
   - ucAlgoID == _goodAlgoID1
   - uszChallenge == _goodChallenge
   - unChallengeLen == _goodChallengeLen
   - uszCryptogram == uszCryptogram2004
   - punCryptogramLen == punCryptogramLen2004.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_BAD_HANDLE or the return code BSI_NO_CARDSERVICE.

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiSkiInternalAuthenticate() call returns the code BSI_BAD_HANDLE, then:

   Perform test for Assertion 9.20.4.1 using
   - hCard == hCard2004.

Print
"gscBsiSkiInternalAuthenticate() (Discovery Method 1, Final Mode) called with a bad handle has been verified.
Status: Test 20.4 Passed."

Case 2: If the gscBsiSkiInternalAuthenticate() call returns the code BSI_NO_CARDSERVICE, then print
"gscBsiSkiInternalAuthenticate() is not supported.
Status: Test 20.4 Not Supported."

Case 3: If the gscBsiSkiInternalAuthenticate() call does not return the code BSI_BAD_HANDLE or the code BSI_NO_CARDSERVICE, then print
"gscBsiSkiInternalAuthenticate() (Discovery Method 1, Final Mode) called with a bad handle returned an incorrect code.
Status: Test 20.4 Failed."

Test for Assertion 20.5

The function is tested with another application having established a transaction lock (Discovery Method 1, Discovery Mode).

Until we encounter implementations that allow multiple simultaneous applications, I don't think we need to worry about this assertion.

Test for Assertion 20.6

The function is tested using a bad AID value (Discovery Method 1, Discovery Mode).

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiSkiInternalAuthenticate().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with AID hCard2006.

3. (Pre) There exists a target SKI provider module on the connected card with the following properties:
   - the ACR for the gscBsiSkiInternalAuthenticate() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by AID value ==
     - _goodGSCAID2, of length _goodGSCAID2Len (GSC), or
     - _goodCACAID2, of length _goodCACAID2Len (CAC).

4. (Pre) There does not exist a container on the connected card with AID value ==
   - _badGSCAID, of length _badGSCAIDLen (GSC), or
   - _badCACAID, of length _badCACAIDLen (CAC).

5. (Pre) Print "Testing of Assertion 20.6".

6. (Pre) Establish an authenticated session with the target SKI provider module on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard2006
   - uszAID == _goodGSCAID2 (GSC) or _goodCACAID2 (CAC)
   - unAIDLen == _goodGSCAID2Len (GSC) or _goodCACAID2Len (CAC)
   - strctAuthenticator == strctAuthenticator2000
   - unAuthNb == 1.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 8.

   Case 2: If the gscBsiUtilAcquireContext() call does not return the code BSI_OK, then print
   "A session cannot be established. Assertion 20.6 of gscBsiSkiInternalAuthenticate() cannot be tested".
   End Test for Assertion 20.6.

7. (Pre) Allocate the unsigned long* punCryptogramLen2006 == 0.

8. Make a gscBsiSkiInternalAuthenticate() call to the SPS, using
   - hCard == hCard2006
   - uszAID == _badGSCAID (GSC) or _badCACAID (CAC)
   - unAIDLen == _badGSCAIDLen (GSC) or _badCACAIDLen (CAC)
   - ucAlgoID == _goodAlgoID1
   - uszChallenge == _goodChallenge
   - unChallengeLen == _goodChallengeLen
   - uszCryptogram == NULL
   - punCryptogramLen == punCryptogramLen2006.

<u>Verification Goal</u>:
To verify the Expected Results:
1. The call returns
   - the return code BSI_BAD_AID or the return code BSI_NO_CARDSERVICE.

<u>Verification and Reporting Scenario</u>:
1. **Case 1**: If the gscBsiSkiInternalAuthenticate() call returns the code BSI_BAD_AID, then:

   Perform test for Assertion 9.20.6.1 using
   - hCard == hCard2006.

   Print
   "gscBsiSkiInternalAuthenticate() (Discovery Method 1, Discovery Mode) called with a bad AID value has been verified.
   Status: Test 20.6 Passed."

   Case 2: If the gscBsiSkiInternalAuthenticate() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiSkiInternalAuthenticate() is not supported.
   Status: Test 20.6 Not Supported."

Case 3: If the gscBsiSkiInternalAuthenticate() call does not return the code BSI_BAD_AID or the code BSI_NO_CARDSERVICE, then print

"gscBsiSkiInternalAuthenticate() (Discovery Method 1, Discovery Mode) called with a bad AID value returned an incorrect code. **Status**: Test 20.6 Failed."

Test for Assertion 20.7

The function is tested using a bad AID length (Discovery Method 1, Discovery Mode).

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiSkiInternalAuthenticate().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with PARAM hCard2007.

3. (Pre) There exists a target SKI provider module on the connected card with the following properties:
   - the ACR for the gscBsiSkiInternalAuthenticate() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by AID value ==
     - _goodGSCAID2, of length _goodGSCAID2Len (GSC), or
     - _goodCACAID2, of length _goodCACAID2Len (CAC).

4. (Pre) Print "Testing of Assertion 20.7".

5. (Pre) Establish an authenticated session with the target SKI provider module on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard2007
   - uszAID == _goodGSCAID2 (GSC) or _goodCACAID2 (CAC)
   - unAIDLen == _goodGSCAID2Len (GSC) or _goodCACAID2Len (CAC)
   - strctAuthenticator == strctAuthenticator2000
   - unAuthNb == 1.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 7.

   Case 2: If the gscBsiUtilAcquireContext() call does not return the code BSI_OK, then print

   "A session cannot be established. Assertion 20.7 of gscBsiSkiInternalAuthenticate() cannot be tested".

   End Test for Assertion 20.7.

6. (Pre) Allocate the unsigned long* punCryptogramLen2007 == 0.

7. Make a gscBsiSkiInternalAuthenticate() call to the SPS, using
   - hCard == hCard2007
   - uszAID == _goodGSCAID2 (GSC) or _goodCACAID2 (CAC)
   - unAIDLen == _badAIDLen

- ucAlgoID == _goodAlgoID1
- uszChallenge == _goodChallenge
- unChallengeLen == _goodChallengeLen
- uszCryptogram == NULL
- punCryptogramLen == punCryptogramLen2007.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_BAD_PARAM or the return code BSI_NO_CARDSERVICE.

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiSkiInternalAuthenticate() call returns the code BSI_BAD_PARAM, then:

   Perform test for Assertion 9.20.7.1 using
   - hCard == hCard2007.

   Print
   "gscBsiSkiInternalAuthenticate() (Discovery Method 1, Discovery Mode) called with a bad AID length has been verified.
   Status: Test 20.7 Passed."

   Case 2: If the gscBsiSkiInternalAuthenticate() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiSkiInternalAuthenticate() is not supported.
   Status: Test 20.7 Not Supported."

   Case 3: If the gscBsiSkiInternalAuthenticate() call does not return the code BSI_BAD_PARAM or the code BSI_NO_CARDSERVICE, then print
   "gscBsiSkiInternalAuthenticate() (Discovery Method 1, Discovery Mode) called with a bad AID length returned an incorrect code.
   Status: Test 20.7 Failed."

Test for Assertion 20.8

The function is tested using a bad challenge length parameter (Discovery Method 1, Discovery Mode).

Instantiation Scenario:
1. (Pre) Construct the Starting State for the testing of gscBsiSkiInternalAuthenticate().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard2008.

3. (Pre) There exists a target SKI provider module on the connected card with the following properties:
   - the ACR for the gscBsiSkiInternalAuthenticate() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by AID value ==
     - _goodGSCAID2, of length _goodGSCAID2Len (GSC), or

- _goodCACAID2, of length _goodCACAID2Len (CAC).

4. (Pre) Print "Testing of Assertion 20.8".

5. (Pre) Establish an authenticated session with the target SKI provider module on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard2008
   - uszAID == _goodGSCAID2 (GSC) or _goodCACAID2 (CAC)
   - unAIDLen == _goodGSCAID2Len (GSC) or _goodCACAID2Len (CAC)
   - strctAuthenticator == strctAuthenticator2000
   - unAuthNb == 1.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 6.

   Case 2: If the gscBsiUtilAcquireContext() call does not return the code BSI_OK, then print
   "A session cannot be established. Assertion 20.8 of gscBsiSkiInternalAuthenticate() cannot be tested".
   End Test for Assertion 20.8.

6. (Pre) Allocate the unsigned long* punCryptogramLen2008 == 0.

7. Make a gscBsiSkiInternalAuthenticate() call to the SPS, using
   - hCard == hCard2008
   - uszAID == _goodGSCAID2 (GSC) or _goodCACAID2 (CAC)
   - unAIDLen == _goodGSCAID2Len (GSC) or _goodCACAID2Len (CAC)
   - ucAlgoID == _goodAlgoID1
   - uszChallenge == _goodChallenge
   - unChallengeLen == _badChallengeLen
   - uszCryptogram == NULL
   - punCryptogram == punCryptogram2008.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_BAD_PARAM or the return code BSI_NO_CARDSERVICE.

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiSkiInternalAuthenticate() call returns the code BSI_BAD_PARAM, then:

   Perform test for Assertion 9.20.8.1 using
   - hCard == hCard2008.

   Print
   "gscBsiSkiInternalAuthenticate() (Discovery Method 1, Discovery Mode) called with a bad challenge length parameter has been verified.
   Status: Test 20.8 Passed."

   Case 2: If the gscBsiSkiInternalAuthenticate() call returns the code BSI_NO_CARDSERVICE, then print

"gscBsiSkiInternalAuthenticate() is not supported.
Status: Test 20.8 Not Supported."

Case 3: If the gscBsiSkiInternalAuthenticate() call does not return the code BSI_BAD_PARAM or the code BSI_NO_CARDSERVICE, then print
"gscBsiSkiInternalAuthenticate() (Discovery Method 1, Discovery Mode) called with a bad challenge length parameter returned an incorrect code.
Status: Test 20.8 Failed."

Test for Assertion 20.9

The function is tested using a bad cryptographic algorithm identifier (Discovery Method 1, Discovery Mode).

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiSkiInternalAuthenticate().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard2009.

3. (Pre) There exists a target SKI provider module on the connected card with the following properties:
   - the ACR for the gscBsiSkiInternalAuthenticate() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by AID value ==
     - _goodGSCAID2, of length _goodGSCAID2Len (GSC), or
     - _goodCACAID2, of length _goodCACAID2Len (CAC).

4. (Pre) Print "Testing of Assertion 20.9".

5. (Pre) Establish an authenticated session with the target SKI provider module on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard2009
   - uszAID == _goodGSCAID2 (GSC) or _goodCACAID2 (CAC)
   - unAIDLen == _goodGSCAID2Len (GSC) or _goodCACAID2Len (CAC)
   - strctAuthenticator == strctAuthenticator2000
   - unAuthNb == 1.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 7.

   Case 2: If the gscBsiUtilAcquireContext() call does not return the code BSI_OK, then print
   "A session cannot be established. Assertion 20.9 of gscBsiSkiInternalAuthenticate() cannot be tested".
   End Test for Assertion 20.9.

6. (Pre) Allocate the unsigned long* punCryptogramLen2009 == 0.

7. Make a gscBsiSkiInternalAuthenticate() call to the SPS, using
    - hCard == hCard2009
    - uszAID == _goodGSCAID2 (GSC) or _goodCACAID2 (CAC)
    - unAIDLen == _goodGSCAID2Len (GSC) or _goodCACAID2Len (CAC)
    - ucAlgoID == _goodAlgoID2
    - uszChallenge == _goodChallenge
    - unChallengeLen == _goodChallengeLen
    - uszCryptogram == NULL
    - punCryptogramLen == punCryptogramLen2009.

Verification Goal:
To verify the Expected Results:
1. The call returns
    - the return code BSI_BAD_ALGO_ID or the return code BSI_NO_CARDSERVICE.

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiSkiInternalAuthenticate() call returns the code BSI_BAD_ALGO_ID, then:

    Perform test for Assertion 9.20.9.1 using
    - hCard == hCard2009.

    Print
    "gscBsiSkiInternalAuthenticate() (Discovery Method 1, Discovery Mode) called with a bad cryptographic algorithm identifier has been verified.
    Status: Test 20.9 Passed."

    Case 2: If the gscBsiSkiInternalAuthenticate() call returns the code BSI_NO_CARDSERVICE, then print
    "gscBsiSkiInternalAuthenticate() is not supported.
    Status: Test 20.9 Not Supported."

    Case 3: If the gscBsiSkiInternalAuthenticate() call does not return the code BSI_BAD_ALGO_ID or the code BSI_NO_CARDSERVICE, then print
    "gscBsiSkiInternalAuthenticate() (Discovery Method 1, Discovery Mode) called with a bad cryptographic algorithm identifier returned an incorrect code.
    Status: Test 20.9 Failed."

Test for Assertion 20.10

The function is tested using a removed card (Discovery Method 1, Discovery Mode).

Instantiation Scenario:
1. (Pre) Construct the Starting State for the testing of gscBsiSkiInternalAuthenticate().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard2010.

3. (Pre) There exists a target SKI provider module on the connected card with the following properties:

- the ACR for the gscBsiSkiInternalAuthenticate() service has the value BSI_ACR_PIN
- the value of the PIN is _PIN
- the container is represented by AID value ==
  - _goodGSCAID2, of length _goodGSCAID2Len (GSC), or
  - _goodCACAID2, of length _goodCACAID2Len (CAC).

4. (Pre) Print "Testing of Assertion 20.10".

5. (Pre) Establish an authenticated session with the target SKI provider module on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard2010
   - uszAID == _goodGSCAID2 (GSC) or _goodCACAID2 (CAC)
   - unAIDLen == _goodGSCAID2Len (GSC) or _goodCACAID2Len (CAC)
   - strctAuthenticator == strctAuthenticator2000
   - unAuthNb == 1.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 6.

   Case 2: If the gscBsiUtilAcquireContext() call does not return the code BSI_OK, then print
   "A session cannot be established. Assertion 20.10 of gscBsiSkiInternalAuthenticate() cannot be tested".
   End Test for Assertion 20.10.

6. (Pre) Allocate the unsigned long* punCryptogramLen2010 == 0.

7. (Pre) Remove the connected card from the reader.

8. Make a gscBsiSkiInternalAuthenticate() call to the SPS, using
   - hCard == hCard2010
   - uszAID == _goodGSCAID2 (GSC) or _goodCACAID2 (CAC)
   - unAIDLen == _goodGSCAID2Len (GSC) or _goodCACAID2Len (CAC)
   - ucAlgoID == _goodAlgoID1
   - uszChallenge == _goodChallenge
   - unChallengeLen == _goodChallengeLen
   - uszCryptogram == NULL
   - punCryptogramLen == punCryptogramLen2010.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_CARD_REMOVED or the return code BSI_NO_CARDSERVICE.

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiSkiInternalAuthenticate() call returns the code BSI_CARD_REMOVED, then print
   "gscBsiSkiInternalAuthenticate() (Discovery Method 1, Discovery Mode) called with the connected card removed has been verified.
   Status: Test 20.10 Passed."

Case 2: If the gscBsiSkiInternalAuthenticate() call returns the code BSI_NO_CARDSERVICE, then print
"gscBsiSkiInternalAuthenticate() is not supported.
Status: Test 20.10 Not Supported."

Case 3: If the gscBsiSkiInternalAuthenticate() call does not return the code BSI_CARD_REMOVED or the code BSI_NO_CARDSERVICE, then print
"gscBsiSkiInternalAuthenticate() (Discovery Method 1, Discovery Mode) called with the connected card removed returned an incorrect code.
Status: Test 20.10 Failed."

Test for Assertion 20.11

The function is tested without fulfilling the applicable ACR (Discovery Method 1, Discovery Mode).

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiSkiInternalAuthenticate().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard2011.

3. (Pre) There exists a target SKI provider module on the connected card with the following properties:
   - the ACR for the gscBsiSkiInternalAuthenticate() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by AID value ==
     - _goodGSCAID2, of length _goodGSCAID2Len (GSC), or
     - _goodCACAID2, of length _goodCACAID2Len (CAC).

4. (Pre) Print "Testing of Assertion 20.11".

5. (Pre) Ensure that there is no authenticated session with the target container:

   Make a gscBsiUtilReleaseContext() call to the SPS, using
   - hCard == hCard2011
   - uszAID == _goodGSCAID2 (GSC) or _goodCACAID2 (CAC)
   - unAIDLen == _goodGSCAID2Len (GSC) or _goodCACAID2Len (CAC).

6. (Pre) Allocate the unsigned long* punCryptogramLen2011 == 0.

7. Make a gscBsiSkiInternalAuthenticate() call to the SPS, using
   - hCard == hCard2011
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - ucAlgoID == _goodAlgoID1
   - uszChallenge == _goodChallenge
   - unChallengeLen == _goodChallengeLen
   - uszCryptogram == NULL
   - punCryptogramLen == punCryptogramLen2011.

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_ACCESS_DENIED or the return code BSI_NO_CARDSERVICE.

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiSkiInternalAuthenticate() call returns the code BSI_ACCESS_DENIED, then:

   Perform test for Assertion 9.20.11.1 using
   - hCard == hCard2011.

   Print
   "gscBsiSkiInternalAuthenticate() (Discovery Method 1, Discovery Mode) called without fulfilling the applicable ACR has been verified.
   Status: Test 20.11 Passed."

   Case 2: If the gscBsiSkiInternalAuthenticate() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiSkiInternalAuthenticate() is not supported.
   Status: Test 20.11 Not Supported."

   Case 3: If the gscBsiSkiInternalAuthenticate() call does not return the code BSI_ACCESS_DENIED or the code BSI_NO_CARDSERVICE, then print
   "gscBsiSkiInternalAuthenticate() (Discovery Method 1, Discovery Mode) called without fulfilling the applicable ACR returned an incorrect code.
   Status: Test 20.11 Failed."

21. gscBsiPkiCompute()

Starting State for Each Test:

1. There exists a BSIAuthenticator array strctAuthenticator2100 with one element, the structure BSIAuthenticator2100. This structure has fields
   - unAccessMethodType2100 == BSI_AM_PIN
   - unkeyIDOrReference2100 == _keyIDOrReference1
   - uszAuthValue2100 == _goodAuthValue1
   - unAuthValueLen2100 == _goodAuthValue1Len.

Test for Assertion 21.1

The function is tested using valid parameters (Discovery Method 1).

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiPkiCompute().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard2101.

3. (Pre) There exists a target PKI provider module on the connected card with the following properties:
   - the ACR for the gscBsiPkiCompute() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by AID value ==
     - _goodGSCAID1, of length _goodGSCAID1Len (GSC), or
     - _goodCACAID1, of length _goodCACAID1Len (CAC).

4. (Pre) Print "Testing of Assertion 21.1".

5. (Pre) Establish an authenticated session with the target PKI provider module on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard2101
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - strctAuthenticator == strctAuthenticator2100
   - unAuthNb == 1.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 6.

   Case 2: If the gscBsiUtilAcquireContext() call does not return the code BSI_OK, then print
   "A session cannot be established. Assertion 21.1 of gscBsiPkiCompute() cannot be tested".
   End Test for Assertion 21.1.

6. (Pre) Allocate the unsigned long* punResultLen2101 == 0.

7. Make a gscBsiPkiCompute() call to the SPS, using
   - hCard == hCard2101
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - ucAlgoID == _goodAlgoID2
   - uszMessage == _goodMessage
   - unMessageLen == _goodMessageLen
   - uszResult == NULL
   - punResultLen == punResultLen2101.

   Case 1: If the gscBsiPkiCompute() call returns the code BSI_OK, then print
   "gscBsiPkiCompute() (Discovery Method 1, Discovery Mode) succeeded."
   Continue with 8.

   Case 2: If the gscBsiPkiCompute() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiPkiCompute() is not supported.
   Status: Test 21.1 Not Supported."
   End Test for Assertion 21.1.

   Case 3: If the gscBsiPkiCompute() call does not return the code BSI_OK or the code BSI_NO_CARDSERVICE, then print
   "gscBsiPkiCompute() (Discovery Method 1, Discovery Mode) returned an incorrect code.
   Status: Test 21.1 Failed."
   End Test for Assertion 21.1.

8. Allocate the string uszResult2101 with length == punResultLen2101*.

9. Make a gscBsiPkiCompute() call to the SPS, using
   - hCard == hCard2101
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - ucAlgoID == _goodAlgoID2
   - uszMessage == _goodMessage
   - unMessageLen == _goodMessageLen
   - uszResult == uszResult2101
   - punResultLen == punResultLen2101.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_OK or the return code BSI_NO_CARDSERVICE.

2. If the return code is BSI_OK, then uszResult2101 == a string containing the signature returned from the connected card.

Perform this verification by inspection.

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiPkiCompute() call returns the code BSI_OK, then manually inspect uszResult2101:

Case 1.1: If uszResult2101 contains a valid signature, then print
"gscBsiPkiCompute() (Discovery Method 1, Final Mode) called with valid parameters has been verified.
Status: Test 21.1 Passed."

Case 1.2: If uszResult2101 does not contain a valid signature, then print
"gscBsiPkiCompute() (Discovery Method 1, Final Mode) called with valid parameters has not been verified.
Status: Test 21.1 Failed."

Case 2: If the gscBsiPkiCompute() call returns the code BSI_NO_CARDSERVICE, then print
"gscBsiPkiCompute() is not supported.
Status: Test 21.1 Not Supported."

Case 3: If the gscBsiPkiCompute() call does not return the code BSI_OK or the code BSI_NO_CARDSERVICE, then print
"gscBsiPkiCompute() (Discovery Method 1, Final Mode) called with valid parameters returned an incorrect code.
Status: Test 21.1 Failed."

Test for Assertion 21.2

The function is tested using valid parameters (Discovery Method 2).

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiPkiCompute().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard2102.

3. (Pre) There exists a target PKI provider module on the connected card with the following properties:
   - the ACR for the gscBsiPkiCompute() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by AID value ==
     - _goodGSCAID1, of length _goodGSCAID1Len (GSC), or
     - _goodCACAID1, of length _goodCACAID1Len (CAC).

4. (Pre) Print "Testing of Assertion 21.2".

5. (Pre) Establish an authenticated session with the target PKI provider module on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard2102
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - strctAuthenticator == strctAuthenticator2100
   - unAuthNb == 1.

Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 6.

Case 2: If the gscBsiUtilAcquireContext() call does not return the code BSI_OK, then print
"A session cannot be established. Assertion 21.2 of gscBsiPkiCompute() cannot be tested".
End Test for Assertion 21.2.

6. (Pre) Allocate the string uszResult2102 with length == 0.

7. (Pre) Allocate the unsigned long* punResultLen2102 == 0.

8. Make a gscBsiPkiCompute() call to the SPS, using
   - hCard == hCard2102
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - ucAlgoID == _goodAlgoID2
   - uszMessage == _goodMessage
   - unMessageLen == _goodMessageLen
   - uszResult == uszResult2102
   - punResultLen == punResultLen2102.

   Case 1: If the gscBsiPkiCompute() call returns the code BSI_INSUFFICIENT_BUFFER, then print
   "gscBsiPkiCompute() (Discovery Method 2, Discovery Mode) succeeded."
   Continue with 9.

   Case 2: If the gscBsiPkiCompute() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiPkiCompute() is not supported.
   Status: Test 21.2 Not Supported."
   End Test for Assertion 21.2.

   Case 3: If the gscBsiPkiCompute() call does not return the code BSI_INSUFFICIENT_BUFFER or the code BSI_NO_CARDSERVICE, then print
   "gscBsiPkiCompute() (Discovery Method 2, Discovery Mode) did not return the code BSI_INSUFFICIENT_BUFFER.
   Status: Test 21.2 Failed."
   End Test for Assertion 21.2.

9. Re-allocate the string uszResult2102 with length == punResultLen2102*.

10. Make a gscBsiPkiCompute() call to the SPS, using
    - hCard == hCard2102
    - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
    - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
    - ucAlgoID == _goodAlgoID2
    - uszMessage == _goodMessage
    - unMessageLen == _goodMessageLen
    - uszResult == uszResult2102
    - punResultLen == punResultLen2102.

Verification Goal:

To verify the Expected Results:

1. The call returns
   - the return code BSI_OK or the return code BSI_NO_CARDSERVICE.

2. If the return code is BSI_OK, then uszResult2102 == a string containing the signature returned from the connected card.

Perform this verification by inspection.

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiPkiCompute() call returns the code BSI_OK, then manually inspect uszResult2102:

   Case 1.1: If uszResult2102 contains a valid signature returned from the connected card, then print
   "gscBsiPkiCompute() (Discovery Method 2, Final Mode) called with valid parameters has been verified.
   Status: Test 21.2 Passed."

   Case 1.2: If uszResult2102 does not contain a valid signature returned from the connected card, then print
   "gscBsiPkiCompute() (Discovery Method 2, Final Mode) called with valid parameters has not been verified.
   Status: Test 21.2 Failed."

   Case 2: If the gscBsiPkiCompute() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiPkiCompute() is not supported.
   Status: Test 21.2 Not Supported."

   Case 3: If the gscBsiPkiCompute() call does not return the code BSI_OK or the code BSI_NO_CARDSERVICE, then print
   "gscBsiPkiCompute() (Discovery Method 2, Final Mode) called with valid parameters returned an incorrect code.
   Status: Test 21.2 Failed."

Test for Assertion 21.3

The function is tested using a bad handle (Discovery Method 1, Discovery Mode).

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiPkiCompute().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard2103.

3. (Pre) There exists a target PKI provider module on the connected card with the following properties:
   - the ACR for the gscBsiPkiCompute() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by AID value ==
     - _goodGSCAID1, of length _goodGSCAID1Len (GSC), or
     - _goodCACAID1, of length _goodCACAID1Len (CAC).

4. (Pre) Print "Testing of Assertion 21.3".

5. (Pre) Establish an authenticated session with the target PKI provider module on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard2103
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - strctAuthenticator == strctAuthenticator2100
   - unAuthNb == 1.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 6.

   Case 2: If the gscBsiUtilAcquireContext() call does not return the code BSI_OK, then print
   "A session cannot be established. Assertion 21.3 of gscBsiPkiCompute() cannot be tested".
   End Test for Assertion 21.3.

6. (Pre) Allocate the unsigned long* punResultLen2103 == 0.

7. Make a gscBsiPkiCompute() call to the SPS, using
   - hCard /= hCard2103
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - ucAlgoID == _goodAlgoID2
   - uszMessage == _goodMessage
   - unMessageLen == _goodMessageLen
   - uszResult == NULL
   - punResultLen == punResultLen2103.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_BAD_HANDLE or the return code BSI_NO_CARDSERVICE.

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiPkiCompute() call returns the code BSI_BAD_HANDLE, then:

   Perform test for Assertion 9.21.3.1 using
   - hCard == hCard2103.

   Print
   "gscBsiPkiCompute() (Discovery Method 1, Discovery Mode) called with a bad handle has been verified.
   Status: Test 21.3 Passed."

   Case 2: If the gscBsiPkiCompute() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiPkiCompute() is not supported.
   Status: Test 21.3 Not Supported."

Case 3: If the gscBsiPkiCompute() call does not return the code BSI_BAD_HANDLE or the code BSI_NO_CARDSERVICE, then print
"gscBsiPkiCompute() (Discovery Method 1, Discovery Mode) called with a bad handle returned an incorrect code.
Status: Test 21.3 Failed."

Test for Assertion 21.4

The function is tested using a bad handle (Discovery Method 4, Final Mode).

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiPkiCompute().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard2104.

3. (Pre) There exists a target PKI provider module on the connected card with the following properties:
   - the ACR for the gscBsiPkiCompute() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by AID value ==
     - _goodGSCAID1, of length _goodGSCAID1Len (GSC), or
     - _goodCACAID1, of length _goodCACAID1Len (CAC).

4. (Pre) Print "Testing of Assertion 21.4".

5. (Pre) Establish an authenticated session with the target PKI provider module on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard2104
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - strctAuthenticator == strctAuthenticator2100
   - unAuthNb == 1.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 6.

   Case 2: If the gscBsiUtilAcquireContext() call does not return the code BSI_OK, then print
   "A session cannot be established. Assertion 21.4 of gscBsiPkiCompute() cannot be tested".
   End Test for Assertion 21.4.

6. (Pre) Allocate the string uszResult2104 with length == 0.

7. (Pre) Allocate the unsigned long* punResultLen2104 == 0.

8. Make a gscBsiPkiCompute() call to the SPS, using
   - hCard == hCard2104

- uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
- unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
- ucAlgoID == _goodAlgoID2
- uszMessage == _goodMessage
- unMessageLen == _goodMessageLen
- uszResult == uszResult2104
- punResultLen == punResultLen2104.

Case 1: If the gscBsiPkiCompute() call returns the code BSI_INSUFFICIENT_BUFFER, then print
"gscBsiPkiCompute() (Discovery Method 2, Discovery Mode) succeeded."
Continue with 9.

Case 2: If the gscBsiPkiCompute() call returns the code BSI_NO_CARDSERVICE, then print
"gscBsiPkiCompute() is not supported.
Status: Test 21.4 Not Supported."
End Test for Assertion 21.4.

Case 3: If the gscBsiPkiCompute() call does not return the code BSI_INSUFFICIENT_BUFFER or the code BSI_NO_CARDSERVICE, then print
"gscBsiPkiCompute() (Discovery Method 2, Discovery Mode) did not return the code BSI_INSUFFICIENT_BUFFER.
Status: Test 21.4 Failed."
End Test for Assertion 21.4.

9. Re-allocate the string uszResult2104 with length == punResultLen2104*.

10. Make a gscBsiPkiCompute() call to the SPS, using
    - hCard /= hCard2104
    - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
    - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
    - ucAlgoID == _goodAlgoID2
    - uszMessage == _goodMessage
    - unMessageLen == _goodMessageLen
    - uszResult == uszResult2104
    - punResultLen == punResultLen2104.

Verification Goal:
To verify the Expected Results:
1. The call returns
    - the return code BSI_BAD_HANDLE or the return code BSI_NO_CARDSERVICE.

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiPkiCompute() call returns the code BSI_BAD_HANDLE, then:

   Perform test for Assertion 9.21.4.1 using
   - hCard == hCard2104.

   Print
   "gscBsiPkiCompute() (Discovery Method 1, Final Mode) called with a bad handle has been verified.

Status: Test 21.4 Passed."

Case 2: If the gscBsiPkiCompute() call returns the code BSI_NO_CARDSERVICE, then print
"gscBsiPkiCompute() is not supported.
Status: Test 21.4 Not Supported."

Case 3: If the gscBsiPkiCompute() call does not return the code BSI_BAD_HANDLE or the code BSI_NO_CARDSERVICE, then print
"gscBsiPkiCompute() (Discovery Method 1, Final Mode) called with a bad handle returned an incorrect code.
Status: Test 21.4 Failed."

Test for Assertion 21.5

The function is tested with another application having established a transaction lock (Discovery Method 1, Discovery Mode).

Until we encounter implementations that allow multiple simultaneous applications, I don't think we need to worry about this assertion.

Test for Assertion 21.6

The function is tested using a bad AID value (Discovery Method 1, Discovery Mode).

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiPkiCompute().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard2106.

3. (Pre) There exists a target PKI provider module on the connected card with the following properties:
   - the ACR for the gscBsiPkiCompute() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by AID value ==
     - _goodGSCAID1, of length _goodGSCAID1Len (GSC), or
     - _goodCACAID1, of length _goodCACAID1Len (CAC).

9. (Pre) There does not exist a container on the connected card with AID value ==
   - _badGSCAID, of length _badGSCAIDLen (GSC), or
   - _badCACAID, of length _badCACAIDLen (CAC).

4. (Pre) Print "Testing of Assertion 21.6".

5. (Pre) Establish an authenticated session with the target PKI provider module on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard2106
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)

- unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
- strctAuthenticator == strctAuthenticator2100
- unAuthNb == 1.

Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 8.

Case 2: If the gscBsiUtilAcquireContext() call does not return the code BSI_OK, then print
"A session cannot be established. Assertion 21.6 of gscBsiPkiCompute() cannot be tested".
End Test for Assertion 21.6.

6. (Pre) Allocate the unsigned long* punResultLen2106 == 0.

7. Make a gscBsiPkiCompute() call to the SPS, using
   - hCard == hCard2106
   - uszAID == _badGSCAID (GSC) or _badCACAID (CAC)
   - unAIDLen == _badGSCAIDLen (GSC) or _badCACAIDLen (CAC)
   - ucAlgoID == _goodAlgoID2
   - uszMessage == _goodMessage
   - unMessageLen == _goodMessageLen
   - uszResult == NULL
   - punResultLen == punResultLen2106.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_BAD_AID or the return code BSI_NO_CARDSERVICE.

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiPkiCompute() call returns the code BSI_BAD_AID, then:

   Perform test for Assertion 9.21.6.1 using
   - hCard == hCard2106.

   Print
   "gscBsiPkiCompute() (Discovery Method 1, Discovery Mode) called with a bad AID value has been verified.
   Status: Test 21.6 Passed."

   Case 2: If the gscBsiPkiCompute() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiPkiCompute() is not supported.
   Status: Test 21.6 Not Supported."

   Case 3: If the gscBsiPkiCompute() call does not return the code BSI_BAD_AID or the code BSI_NO_CARDSERVICE, then print
   "gscBsiPkiCompute() (Discovery Method 1, Discovery Mode) called with a bad AID value returned an incorrect code.
   Status: Test 21.6 Failed."

Test for Assertion 21.7

The function is tested using a bad AID length (Discovery Method 1, Discovery Mode).

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiPkiCompute().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard2107.

3. (Pre) There exists a target PKI provider module on the connected card with the following properties:
   - the ACR for the gscBsiPkiCompute() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by AID value ==
     - _goodGSCAID1, of length _goodGSCAID1Len (GSC), or
     - _goodCACAID1, of length _goodCACAID1Len (CAC).

4. (Pre) Print "Testing of Assertion 21.7".

5. (Pre) Establish an authenticated session with the target PKI provider module on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard2107
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - strctAuthenticator == strctAuthenticator2100
   - unAuthNb == 1.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 7.

   Case 2: If the gscBsiUtilAcquireContext() call does not return the code BSI_OK, then print
   "A session cannot be established. Assertion 21.7 of gscBsiPkiCompute() cannot be tested".
   End Test for Assertion 21.7.

6. (Pre) Allocate the unsigned long* punResultLen2107 == 0.

7. Make a gscBsiPkiCompute() call to the SPS, using
   - hCard == hCard2107
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _badAIDLen
   - ucAlgoID == _goodAlgoID2
   - uszMessage == _goodMessage
   - unMessageLen == _goodMessageLen
   - uszResult == NULL
   - punResultLen == punResultLen2107.

Verification Goal:

To verify the Expected Results:

1. The call returns

- the return code BSI_BAD_PARAM or the return code BSI_NO_CARDSERVICE.

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiPkiCompute() call returns the code BSI_BAD_PARAM, then:

   Perform test for Assertion 9.21.7.1 using
   - hCard == hCard2107.

   Print
   "gscBsiPkiCompute() (Discovery Method 1, Discovery Mode) called with a bad AID length has been verified.
   Status: Test 21.7 Passed."

   Case 2: If the gscBsiPkiCompute() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiPkiCompute() is not supported.
   Status: Test 21.7 Not Supported."

   Case 3: If the gscBsiPkiCompute() call does not return the code BSI_BAD_PARAM or the code BSI_NO_CARDSERVICE, then print
   "gscBsiPkiCompute() (Discovery Method 1, Discovery Mode) called with a bad AID length returned an incorrect code.
   Status: Test 21.7 Failed."

Test for Assertion 21.8

The function is tested using a bad message length parameter (Discovery Method 1, Discovery Mode).

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiPkiCompute().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard2108.

3. (Pre) There exists a target PKI provider module on the connected card with the following properties:
   - the ACR for the gscBsiPkiCompute() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by AID value ==
     - _goodGSCAID1, of length _goodGSCAID1Len (GSC), or
     - _goodCACAID1, of length _goodCACAID1Len (CAC).

4. (Pre) Print "Testing of Assertion 21.8".

5. (Pre) Establish an authenticated session with the target PKI provider module on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard2108
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)

- unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
- strctAuthenticator == strctAuthenticator2100
- unAuthNb == 1.

Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 6.

Case 2: If the gscBsiUtilAcquireContext() call does not return the code BSI_OK, then print
"A session cannot be established. Assertion 21.8 of gscBsiPkiCompute() cannot be tested".
End Test for Assertion 21.8.

6. (Pre) Allocate the unsigned long* punResultLen2108 == 0.

7. Make a gscBsiPkiCompute() call to the SPS, using
   - hCard /= hCard2108
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - ucAlgoID == _goodAlgoID2
   - uszMessage == _goodMessage
   - unMessageLen == _badMessageLen
   - uszResult == NULL
   - punResult == punResult2108.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_BAD_PARAM or the return code BSI_NO_CARDSERVICE.

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiPkiCompute() call returns the code BSI_BAD_PARAM, then:

   Perform test for Assertion 9.21.8.1 using
   - hCard == hCard2108.

   Print
   "gscBsiPkiCompute() (Discovery Method 1, Discovery Mode) called with a bad message length parameter has been verified.
   Status: Test 21.8 Passed."

   Case 2: If the gscBsiPkiCompute() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiPkiCompute() is not supported.
   Status: Test 21.8 Not Supported."

   Case 3: If the gscBsiPkiCompute() call does not return the code BSI_BAD_PARAM or the code BSI_NO_CARDSERVICE, then print
   "gscBsiPkiCompute() (Discovery Method 1, Discovery Mode) called with a bad message length parameter returned an incorrect code.
   Status: Test 21.8 Failed."

Test for Assertion 21.9

The function is tested using a bad cryptographic algorithm identifier (Discovery Method 1, Discovery Mode).

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiPkiCompute().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard2109.

3. (Pre) There exists a target PKI provider module on the connected card with the following properties:
   - the ACR for the gscBsiPkiCompute() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by AID value ==
     - _goodGSCAID1, of length _goodGSCAID1Len (GSC), or
     - _goodCACAID1, of length _goodCACAID1Len (CAC).

4. (Pre) Print "Testing of Assertion 21.9".

5. (Pre) Establish an authenticated session with the target PKI provider module on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard2109
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - strctAuthenticator == strctAuthenticator2100
   - unAuthNb == 1.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 6.

   Case 2: If the gscBsiUtilAcquireContext() call does not return the code BSI_OK, then print
   "A session cannot be established. Assertion 21.9 of gscBsiPkiCompute() cannot be tested".
   End Test for Assertion 21.9.

6. (Pre) Allocate the unsigned long* punResultLen2109 == 0.

7. Make a gscBsiPkiCompute() call to the SPS, using
   - hCard == hCard2109
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - ucAlgoID == _goodAlgoID1
   - uszMessage == _goodMessage
   - unMessageLen == _goodMessageLen
   - uszResult == NULL
   - punResultLen == punResultLen2109.

Verification Goal:

To verify the Expected Results:

1. The call returns

- the return code BSI_BAD_ALGO_ID or the return code BSI_NO_CARDSERVICE.

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiPkiCompute() call returns the code BSI_BAD_ALGO_ID, then:

   Perform test for Assertion 9.21.9.1 using
   - hCard == hCard2109.

   Print
   "gscBsiPkiCompute() (Discovery Method 1, Discovery Mode) called with a bad cryptographic algorithm identifier has been verified.
   Status: Test 21.9 Passed."

   Case 2: If the gscBsiPkiCompute() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiPkiCompute() is not supported.
   Status: Test 21.9 Not Supported."

   Case 3: If the gscBsiPkiCompute() call does not return the code BSI_BAD_ALGO_ID or the code BSI_NO_CARDSERVICE, then print
   "gscBsiPkiCompute() (Discovery Method 1, Discovery Mode) called with a bad cryptographic algorithm identifier returned an incorrect code.
   Status: Test 21.9 Failed."

Test for Assertion 21.10

The function is tested using a removed card (Discovery Method 1, Discovery Mode).

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiPkiCompute().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard2110.

3. (Pre) There exists a target PKI provider module on the connected card with the following properties:
   - the ACR for the gscBsiPkiCompute() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by AID value ==
     - _goodGSCAID1, of length _goodGSCAID1Len (GSC), or
     - _goodCACAID1, of length _goodCACAID1Len (CAC).

4. (Pre) Print "Testing of Assertion 21.10".

5. (Pre) Establish an authenticated session with the target PKI provider module on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard2110

- uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
- unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
- strctAuthenticator == strctAuthenticator2100
- unAuthNb == 1.

Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 6.

Case 2: If the gscBsiUtilAcquireContext() call does not return the code BSI_OK, then print
"A session cannot be established. Assertion 21.10 of gscBsiPkiCompute() cannot be tested".
End Test for Assertion 21.10.

6. (Pre) Allocate the unsigned long* punResultLen2110 == 0.

7. (Pre) Remove the connected card from the reader.

8. Make a gscBsiPkiCompute() call to the SPS, using
   - hCard == hCard2110
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - ucAlgoID == _goodAlgoID2
   - uszMessage == _goodMessage
   - unMessageLen == _goodMessageLen
   - uszResult == NULL
   - punResultLen == punResultLen2110.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_CARD_REMOVED or the return code BSI_NO_CARDSERVICE.

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiPkiCompute() call returns the code BSI_CARD_REMOVED, then print
   "gscBsiPkiCompute() (Discovery Method 1, Discovery Mode) called with the connected card removed has been verified.
   Status: Test 21.10 Passed."

   Case 2: If the gscBsiPkiCompute() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiPkiCompute() is not supported.
   Status: Test 21.10 Not Supported."

   Case 3: If the gscBsiPkiCompute() call does not return the code BSI_CARD_REMOVED or the code BSI_NO_CARDSERVICE, then print
   "gscBsiPkiCompute() (Discovery Method 1, Discovery Mode) called with the connected card removed returned an incorrect code.
   Status: Test 21.10 Failed."

Test for Assertion 21.11

The function is tested without fulfilling the applicable ACR (Discovery Method 1, Discovery Mode).

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiPkiCompute().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard2111.

3. (Pre) There exists a target PKI provider module on the connected card with the following properties:
   - the ACR for the gscBsiPkiCompute() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - the container is represented by AID value ==
     - _goodGSCAID1, of length _goodGSCAID1Len (GSC), or
     - _goodCACAID1, of length _goodCACAID1Len (CAC).

4. (Pre) Print "Testing of Assertion 21.11".

5. (Pre) Ensure that there is no authenticated session with the target container:

   Make a gscBsiUtilReleaseContext() call to the SPS, using
   - hCard == hCard2111
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC).

6. (Pre) Allocate the unsigned long* punResultLen2111 == 0.

7. Make a gscBsiPkiCompute() call to the SPS, using
   - hCard == hCard2111
   - uszAID == _goodGSCAID2 (GSC) or _goodCACAID2 (CAC)
   - unAIDLen == _goodGSCAID2Len (GSC) or _goodCACAID2Len (CAC)
   - ucAlgoID == _goodAlgoID2
   - uszMessage == _goodMessage
   - unMessageLen == _goodMessageLen
   - uszResult == NULL
   - punResultLen == punResultLen2111.

Verification Goal:

To verify the Expected Results:

1. The call returns
   - the return code BSI_ACCESS_DENIED or the return code BSI_NO_CARDSERVICE.

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiPkiCompute() call returns the code BSI_ACCESS_DENIED, then:

   Perform test for Assertion 9.21.11.1 using
   - hCard == hCard2111.

   Print

"gscBsiPkiCompute() (Discovery Method 1, Discovery Mode) called without fulfilling the applicable ACR has been verified.
Status: Test 21.11 Passed."

Case 2: If the gscBsiPkiCompute() call returns the code BSI_NO_CARDSERVICE, then print
"gscBsiPkiCompute() is not supported.
Status: Test 21.11 Not Supported."

Case 3: If the gscBsiPkiCompute() call does not return the code BSI_ACCESS_DENIED or the code BSI_NO_CARDSERVICE, then print
"gscBsiPkiCompute() (Discovery Method 1, Discovery Mode) called without fulfilling the applicable ACR returned an incorrect code.
Status: Test 21.11 Failed."

22. gscBsiPkiGetCertificate()

Starting State for Each Test:

1. There exists a BSIAuthenticator array strctAuthenticator2200 with one element, the structure BSIAuthenticator2200. This structure has fields
   - unAccessMethodType2200 == BSI_AM_PIN
   - unkeyIDOrReference2200 == _keyIDOrReference1
   - uszAuthValue2200 == _goodAuthValue1
   - unAuthValueLen2200 == _goodAuthValue1Len.

Test for Assertion 22.1

The function is tested using valid parameters (Discovery Method 1).

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiPkiGetCertificate().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard2201.

3. (Pre) There exists a target PKI provider container on the connected card with AID value ==
   - _goodGSCAID1, of length _goodGSCAID1Len (GSC), or
   - _goodCACAID1, of length _goodCACAID1Len (CAC).

4. (Pre) Print "Testing of Assertion 22.1".

5. (Pre) Establish an authenticated session with the target PKI provider module on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard2201
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - strctAuthenticator == strctAuthenticator2200
   - unAuthNb == 1.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 6.

   Case 2: If the gscBsiUtilAcquireContext() call does not return the code BSI_OK, then print
   "A session cannot be established. Assertion 22.1 of gscBsiSkiInternalAuthenticate() cannot be tested".
   End Test for Assertion 22.1.

6. (Pre) Allocate the unsigned long* punCertificateLen2201 == 0.

7. Make a gscBsiPkiGetCertificate() call to the SPS, using
   - hCard == hCard2201
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)

- uszCertificate == NULL
- punCertificateLen == punCertificateLen2201.

Case 1: If the gscBsiPkiGetCertificate() call returns the code BSI_OK, then print
"gscBsiPkiGetCertificate() (Discovery Method 1, Discovery Mode) succeeded."
Continue with 8.

Case 2: If the gscBsiPkiGetCertificate() call returns the code BSI_NO_CARDSERVICE, then print
"gscBsiPkiGetCertificate() is not supported.
Status: Test 22.1 Not Supported."
End Test for Assertion 22.1.

Case 3: If the gscBsiPkiGetCertificate() call does not return the code BSI_OK or the code BSI_NO_CARDSERVICE, then print
"gscBsiPkiGetCertificate() (Discovery Method 1, Discovery Mode) returned an incorrect code.
Status: Test 22.1 Failed."
End Test for Assertion 22.1.

8. Allocate the unsigned string uszCertificate2201 with length == punCertificateLen2201*.

9. Make a gscBsiPkiGetCertificate() call to the SPS, using
   - hCard == hCard2201
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - uszCertificate == uszCertificate2201
   - punCertificateLen == punCertificateLen2201.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_OK or the return code BSI_NO_CARDSERVICE.

2. If the return code is BSI_OK, then uszCertificate2201 == a string containing the certificate returned from the connected card.

Perform this verification by inspection.

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiPkiGetCertificate() call returns the code BSI_OK, then manually inspect the string uszCertificate2201.

   Case 1.1: If uszCertificate2201 contains a valid certificate, then print
   "gscBsiPkiGetCertificate() (Discovery Method 1, Final Mode) called with valid parameters has been verified by inspection.
   Status: Test 22.1 Passed."

   Case 1.2: If uszCertificate2201 does not contain a valid certificate, then print
   "gscBsiPkiGetCertificate() (Discovery Method 1, Final Mode) called with valid parameters has not been verified by inspection.

Status: Test 22.1 Failed."

Case 2: If the gscBsiPkiGetCertificate() call returns the code BSI_NO_CARDSERVICE, then print
"gscBsiPkiGetCertificate() is not supported.
Status: Test 22.1 Not Supported."

Case 3: If the gscBsiPkiGetCertificate() call does not return the code BSI_OK or the code BSI_NO_CARDSERVICE, then print
"gscBsiPkiGetCertificate() (Discovery Method 1, Final Mode) called with valid parameters returned an incorrect code.
Status: Test 22.1 Failed."

Test for Assertion 22.2

The function is tested using valid parameters (Discovery Method 2).

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiPkiGetCertificate().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard2202.

3. (Pre) There exists a target PKI provider container on the connected card with AID value ==
   - _goodGSCAID1, of length _goodGSCAID1Len (GSC), or
   - _goodCACAID1, of length _goodCACAID1Len (CAC).

4. (Pre) Print "Testing of Assertion 22.2".

5. (Pre) Establish an authenticated session with the target PKI provider module on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard2202
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - strctAuthenticator == strctAuthenticator2200
   - unAuthNb == 1.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 5.

   Case 2: If the gscBsiUtilAcquireContext() call does not return the code BSI_OK, then print
   "A session cannot be established. Assertion 22.2 of gscBsiSkiInternalAuthenticate() cannot be tested".
   End Test for Assertion 22.2.

6. (Pre) Allocate the string uszCertificate2202 with length == 0.

7. (Pre) Allocate the unsigned long* punCertificateLen2202 == 0.

8. Make a gscBsiPkiGetCertificate() call to the SPS, using

- hCard == hCard2202
- uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
- unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
- uszCertificate == uszCertificate2202
- punCertificateLen == punCertificateLen2202.

Case 1: If the gscBsiPkiGetCertificate() call returns the code BSI_INSUFFICIENT_BUFFER, then print
"gscBsiPkiGetCertificate() (Discovery Method 2, Discovery Mode) correctly returned the code BSI_INSUFFICIENT_BUFFER"
Continue with 7.

Case 2: If the gscBsiPkiGetCertificate() call returns the code BSI_NO_CARDSERVICE, then print
"gscBsiPkiGetCertificate() is not supported.
Status: Test 22.2 Not Supported."
End Test for assertion 22.2.

Case 3: If the gscBsiPkiGetCertificate() call does not return the code BSI_INSUFFICIENT_BUFFER or the code BSI_NO_CARDSERVICE, then print
"gscBsiPkiGetCertificate() (Discovery Method 2, Discovery Mode) did not return the code BSI_INSUFFICIENT_BUFFER.
Status: Test 22.2 Failed."
End Test for Assertion 22.2.

9. Re-allocate the string uszCertificate2202 with length == punCertificateLen2202*.

10. Make a gscBsiPkiGetCertificate() call to the SPS, using
    - hCard == hCard2202
    - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
    - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
    - uszCertificate == uszCertificate2202
    - punCertificateLen == punCertificateLen2202.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_OK or the return code BSI_NO_CARDSERVICE.

2. If the return code is BSI_OK, then uszCertificate2202 == a string containing the certificate returned from the connected card.

Perform this verification by inspection.

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiPkiGetCertificate() call returns the code BSI_OK, then manually inspect the string uszCertificate2202.

   Case 1.1: If uszCertificate2202 contains a valid certificate, then print
   "gscBsiPkiGetCertificate() (Discovery Method 2, Final Mode) called with valid parameters has been verified by inspection.
   Status: Test 22.2 Passed."

Case 1.2: If uszCertificate2202 does not contain a valid certificate, then print
"gscBsiPkiGetCertificate() (Discovery Method 2, Final Mode) called with valid parameters has not been verified by inspection.
Status: Test 22.2 Failed."

Case 2: If the gscBsiPkiGetCertificate() call returns the code BSI_NO_CARDSERVICE, then print
"gscBsiPkiGetCertificate() is not supported.
Status: Test 22.2 Not Supported."

Case 3: If the gscBsiPkiGetCertificate() call does not return the code BSI_OK or the code BSI_NO_CARDSERVICE, then print
"gscBsiPkiGetCertificate() (Discovery Method 2, Final Mode) called with valid parameters returned an incorrect code.
Status: Test 22.2 Failed."

Test for Assertion 22.3

The function is tested using a bad handle (Discovery Method 1, Discovery Mode).

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiPkiGetCertificate().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard2203.

3. (Pre) There exists a target PKI provider container on the connected card with AID value ==
   - _goodGSCAID1, of length _goodGSCAID1Len (GSC), or
   - _goodCACAID1, of length _goodCACAID1Len (CAC).

4. (Pre) Print "Testing of Assertion 22.3".

5. (Pre) Establish an authenticated session with the target PKI provider module on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard2203
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - strctAuthenticator == strctAuthenticator2200
   - unAuthNb == 1.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 6.

   Case 2: If the gscBsiUtilAcquireContext() call does not return the code BSI_OK, then print
   "A session cannot be established. Assertion 22.3 of gscBsiSkiInternalAuthenticate() cannot be tested".
   End Test for Assertion 22.3.

6. (Pre) Allocate the unsigned long* punCertificateLen2203 == 0.

7. Make a gscBsiPkiGetCertificate() call to the SPS, using
   - hCard /= hCard2203
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - uszCertificate == NULL
   - punCertificateLen == punCertificateLen2203.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_BAD_HANDLE or the return code BSI_NO_CARDSERVICE.

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiPkiGetCertificate() call returns the code BSI_BAD_HANDLE, then:

   Perform Test for Assertion 9.22.3.1 using
   - hCard == hCard2203.

   Print
   "gscBsiPkiGetCertificate() (Discovery Method 1, Discovery Mode) called with a bad handle has been verified.
   Status: Test 22.3 Passed."

   Case 2: If the gscBsiPkiGetCertificate() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiPkiGetCertificate() is not supported.
   Status: Test 22.3 Not Supported."

   Case 3: If the gscBsiPkiGetCertificate() call does not return the code BSI_BAD_HANDLE or the code BSI_NO_CARDSERVICE, then print
   "gscBsiPkiGetCertificate() (Discovery Method 1, Discovery Mode) called with a bad handle returned an incorrect code.
   Status: Test 22.3 Failed."

Test for Assertion 22.4

The function is tested using a bad handle (Discovery Method 1, Final Mode).

Instantiation Scenario:
1. (Pre) Construct the Starting State for the testing of gscBsiPkiGetCertificate().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard2203.

3. (Pre) There exists a target PKI provider container on the connected card with AID value ==
   - _goodGSCAID1, of length _goodGSCAID1Len (GSC), or
   - _goodCACAID1, of length _goodCACAID1Len (CAC).

4. (Pre) Print "Testing of Assertion 22.4".

5. (Pre) Establish an authenticated session with the target PKI provider module on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard2204
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - strctAuthenticator == strctAuthenticator2200
   - unAuthNb == 1.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 6.

   Case 2: If the gscBsiUtilAcquireContext() call does not return the code BSI_OK, then print
   "A session cannot be established. Assertion 22.4 of gscBsiSkiInternalAuthenticate() cannot be tested".
   End Test for Assertion 22.4.

6. (Pre) Allocate the unsigned long* punCertificateLen2204 == 0.

7. Make a gscBsiPkiGetCertificate() call to the SPS, using
   - hCard == hCard2204
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - uszCertificate == NULL
   - punCertificateLen == punCertificateLen2204.

   Case 1: If the gscBsiPkiGetCertificate() call returns the code BSI_OK, then print
   "gscBsiPkiGetCertificate() (Discovery Method 1, Discovery Mode) succeeded."
   Continue with 8.

   Case 2: If the gscBsiPkiGetCertificate() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiPkiGetCertificate() is not supported.
   Status: Test 22.4 Not Supported."
   End Test for Assertion 22.2.

   Case 3: If the gscBsiPkiGetCertificate() call does not return the code BSI_OK or the code BSI_NO_CARDSERVICE, then print
   "gscBsiPkiGetCertificate() (Discovery Method 1, Discovery Mode) did not return the code BSI_OK.
   Status: Test 22.4 Failed."
   End Test for Assertion 22.4.

8. Allocate the string uszCertificate2204 with length == punCertificateLen2204*.

9. Make a gscBsiPkiGetCertificate() call to the SPS, using
   - hCard /= hCard2204
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)

- uszCertificate == uszCertificate2204
- punCertificateLen == punCertificateLen2204.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_BAD_HANDLE or the code BSI_NO_CARDSERVICE.

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiPkiGetCertificate() call returns the code BSI_BAD_HANDLE, then:

   Perform Test for Assertion 9.22.4.1

   Print
   "gscBsiPkiGetCertificate() (Discovery Method 1, Final Mode) called with a bad handle has been verified.
   Status: Test 22.4 Passed."

   Case 2: If the gscBsiPkiGetCertificate() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiPkiGetCertificate() is not supported.
   Status: Test 22.4 Not Supported."

   Case 3: If the gscBsiPkiGetCertificate() call does not return the code BSI_BAD_HANDLE or the code BSI_NO_CARDSERVICE, then print
   "gscBsiPkiGetCertificate() (Discovery Method 1, Final Mode) called with a bad handle returned an incorrect code.
   Status: Test 22.4 Failed."

Test for Assertion 22.5

The function is tested with another application having established a transaction lock (Discovery Method 1, Discovery Mode).

Until we encounter implementations that allow multiple simultaneous applications, I don't think we need to worry about this assertion.

Test for Assertion 22.6

The function is tested using a bad AID value (Discovery Method 1, Discovery Mode).

Instantiation Scenario:
1. (Pre) Construct the Starting State for the testing of gscBsiPkiGetCertificate().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard2206.

3. (Pre) There exists a target PKI provider container on the connected card with AID value ==
   - _goodGSCAID1, of length _goodGSCAID1Len (GSC), or
   - _goodCACAID1, of length _goodCACAID1Len (CAC).

4. (Pre) There does not exist a container on the connected card with AID value ==
   - _badGSCAID, of length _badGSCAIDLen (GSC), or
   - _badCACAID, of length _badCACAIDLen (CAC).

5. (Pre) Print "Testing of Assertion 22.6".

6. (Pre) Establish an authenticated session with the target PKI provider module on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard2206
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - strctAuthenticator == strctAuthenticator2200
   - unAuthNb == 1.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 7.

   Case 2: If the gscBsiUtilAcquireContext() call does not return the code BSI_OK, then print
   "A session cannot be established. Assertion 22.6 of gscBsiSkiInternalAuthenticate() cannot be tested".
   End Test for Assertion 22.6.

7. (Pre) Allocate the unsigned long* punCertificateLen2206 == 0.

8. Make a gscBsiPkiGetCertificate() call to the SPS, using
   - hCard == hCard2206
   - uszAID == _badGSCAID (GSC) or _badCACAID (CAC)
   - unAIDLen == _badGSCAIDLen (GSC) or _badCACAIDLen (CAC)
   - uszCertificate == NULL
   - punCertificateLen == punCertificateLen2206.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_BAD_AID or the return code BSI_NO_CARDSERVICE.

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiPkiGetCertificate() call returns the code BSI_BAD_AID, then:

   Perform Test for Assertion 9.22.6.1 using
   - hCard == hCard2206.

   Print
   "gscBsiPkiGetCertificate() (Discovery Method 1, Discovery Mode) called with a bad AID value has been verified.
   Status: Test 22.6 Passed."

   Case 2: If the gscBsiPkiGetCertificate() call returns the code BSI_NO_CARDSERVICE, then print

"gscBsiPkiGetCertificate() is not supported.
Status: Test 22.6 Not Supported."

Case 3: If the gscBsiPkiGetCertificate() call does not return the code BSI_BAD_AID or the code BSI_NO_CARDSERVICE, then print
"gscBsiPkiGetCertificate() (Discovery Method 1, Discovery Mode) called with a bad AID value returned an incorrect code.
Status: Test 22.6 Failed."

Test for Assertion 22.7

The function is tested using a bad AID length (Discovery Method 1, Discovery Mode).

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiPkiGetCertificate().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard2207.

3. (Pre) There exists a target PKI provider container on the connected card with AID value ==
   - _goodGSCAID1, of length _goodGSCAID1Len (GSC), or
   - _goodCACAID1, of length _goodCACAID1Len (CAC).

4. ((Pre) Print "Testing of Assertion 22.7".

5. (Pre) Establish an authenticated session with the target PKI provider module on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard2207
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - strctAuthenticator == strctAuthenticator2200
   - unAuthNb == 1.

   Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 7.

   Case 2: If the gscBsiUtilAcquireContext() call does not return the code BSI_OK, then print
   "A session cannot be established. Assertion 22.7 of gscBsiSkiInternalAuthenticate() cannot be tested".
   End Test for Assertion 22.7.

6. (Pre) Allocate the unsigned long* punCertificateLen2207 == 0.

7. Make a gscBsiPkiGetCertificate() call to the SPS, using
   - hCard == hCard2207
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _badAIDLen
   - uszCertificate == NULL
   - punCertificateLen == punCertificateLen2207.

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_BAD_PARAM or the return code BSI_NO_CARDSERVICE.

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiPkiGetCertificate() call returns the code BSI_BAD_PARAM, then:

   Perform Test for Assertion 9.22.7.1 using
   - hCard == hCard2207.

   Print
   "gscBsiPkiGetCertificate() (Discovery Method 1, Discovery Mode) called with a bad AID length has been verified.
   Status: Test 22.7 Passed."

   Case 2: If the gscBsiPkiGetCertificate() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiPkiGetCertificate() is not supported.
   Status: Test 22.7 Not Supported."

   Case 3: If the gscBsiPkiGetCertificate() call does not return the code BSI_BAD_PARAM or the code BSI_NO_CARDSERVICE, then print
   "gscBsiPkiGetCertificate() (Discovery Method 1, Discovery Mode) called with a bad AID length returned an incorrect code.
   Status: Test 22.7 Failed."

Test for Assertion 22.8

The function is tested with a removed card (Discovery Method 1, Discovery Mode).

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiPkiGetCertificate().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard2208.

3. (Pre) There exists a target PKI provider container on the connected card with AID value ==
   - _goodGSCAID1, of length _goodGSCAID1Len (GSC), or
   - _goodCACAID1, of length _goodCACAID1Len (CAC).

4. (Pre) Print "Testing of Assertion 22.8".

5. (Pre) Establish an authenticated session with the target PKI provider module on the card:

   Make a gscBsiUtilAcquireContext() call to the SPS, using
   - hCard == hCard2208
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)

- unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
- strctAuthenticator == strctAuthenticator2200
- unAuthNb == 1.

Case 1: If the gscBsiUtilAcquireContext() call returns the code BSI_OK, then continue with 6.

Case 2: If the gscBsiUtilAcquireContext() call does not return the code BSI_OK, then print
"A session cannot be established. Assertion 22.8 of gscBsiSkiInternalAuthenticate() cannot be tested".
End Test for Assertion 22.8.

6. (Pre) Allocate the unsigned long* punCertificateLen2208 == 0.

7. Remove the connected card from the reader,

8. Make a gscBsiPkiGetCertificate() call to the SPS, using
   - hCard == hCard2208
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - uszCertificate == NULL
   - punCertificateLen == punCertificateLen2208.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_CARD_REMOVED or the return code BSI_NO_CARDSERVICE.

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiPkiGetCertificate() call returns the code BSI_CARD_REMOVED, then:

   Print
   "gscBsiPkiGetCertificate() (Discovery Method 1, Discovery Mode) called with the connected card removed has been verified.
   Status: Test 22.8 Passed."

   Case 2: If the gscBsiPkiGetCertificate() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiPkiGetCertificate() is not supported.
   Status: Test 22.8 Not Supported."

   Case 3: If the gscBsiPkiGetCertificate() call does not return the code BSI_CARD_REMOVED or the code BSI_NO_CARDSERVICE, then print
   "gscBsiPkiGetCertificate() (Discovery Method 1, Discovery Mode) called with the connected card removed returned an incorrect code.
   Status: Test 22.8 Failed."

Test for Assertion 22.9

The function is tested without fulfilling the applicable ACR (Discovery Method 1, Discovery Mode).

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiPkiGetCertificate().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard2209.

3. (Pre) There exists a target container on the connected card with the following properties:
   - the ACR for the gscBsiPkiCompute() service has the value BSI_ACR_PIN
   - the value of the PIN is _PIN
   - this container is represented by AID value ==
     - _goodGSCAID1, of length _goodGSCAID1Len (GSC), or
     - _goodCACAID1, of length _goodCACAID1Len (CAC).

4. (Pre) Print "Testing of Assertion 22.9".

5. (Pre) Ensure that there is no authenticated session with the target container:

   Make a gscBsiUtilReleaseContext() call to the SPS, using
   - hCard == hCard2209
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)

6. (Pre) Allocate the unsigned long* punCertificateLen2209 == 0.

7. Make a gscBsiPkiGetCertificate() call to the SPS, using
   - hCard == hCard2209
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - uszCertificate == NULL
   - punCertificateLen == punCertificateLen2209.

Verification Goal:

To verify the Expected Results:

1. The call returns
   - the return code BSI_ACCESS_DENIED or the return code BSI_NO_CARDSERVICE.

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiPkiGetCertificate() call returns the code BSI_ACCESS_DENIED, then:

   Perform Test for Assertion 9.22.9.1 using
   - hCard == hCard2209.

   Print
   "gscBsiPkiGetCertificate() (Discovery Method 1, Discovery Mode) called without fulfilling the applicable ACR has been verified.
   Status: Test 22.9 Passed."

Case 2: If the gscBsiPkiGetCertificate() call returns the code BSI_NO_CARDSERVICE, then print
"gscBsiPkiGetCertificate() is not supported.
Status: Test 22.9 Not Supported."

Case 3: If the gscBsiPkiGetCertificate() call does not return the code BSI_ACCESS_DENIED or the code BSI_NO_CARDSERVICE, then print
"gscBsiPkiGetCertificate() (Discovery Method 1, Discovery Mode) called without fulfilling the applicable ACR returned an incorrect code.
Status: Test 22.9 Failed."

23. gscBsiGetCryptoProperties()

Starting state for each Test:

1. There exists a CRYPTOacr variable strctCRYPTOacr2300, with members
   - BSIAcr strctGetChallengeACR2300
     - unsigned long unGetChallengeACRType2300
     - unsigned long unGetChallengeKeyIDOrReference2300
     - unsigned long unGetChallengeAuthID2300
     - unsigned long unGetChallengeACRID2300
   - BSIAcr strctInternalAuthenticateACR2300
     - unsigned long unInternalAuthenticateACRType2300
     - unsigned long unInternalAuthenticateKeyIDOrReference2300
     - unsigned long unInternalAuthenticateAuthID2300
     - unsigned long unInternalAuthenticateACRID2300
   - BSIAcr strctPkiComputeACR2300
     - unsigned long unPkiComputeACRType2300
     - unsigned long unPkiComputeKeyIDOrReference2300
     - unsigned long unPkiComputeAuthID2300
     - unsigned long unPkiComputeACRID2300
   - BSIAcr strctCreateACR2300
     - unsigned long unCreateACRType2300
     - unsigned long unCreateKeyIDOrReference2300
     - unsigned long unCreateAuthID2300
     - unsigned long unCreateACRID2300
   - BSIAcr strctDeleteACR2300
     - unsigned long unDeleteACRType2300
     - unsigned long unDeleteKeyIDOrReference2300
     - unsigned long unDeleteAuthID2300
     - unsigned long unDeleteACRID2300
   - BSIAcr strctReadTagListACR2300
     - unsigned long unReadTagListACRType2300
     - unsigned long unReadTagListKeyIDOrReference2300
     - unsigned long unReadTagListAuthID2300
     - unsigned long unReadTagListACRID2300
   - BSIAcr strctReadValueACR2300
     - unsigned long unReadValueACRType2300
     - unsigned long unReadValueKeyIDOrReference2300
     - unsigned long unReadValueAuthID2300
     - unsigned long unReadValueACRID2300
   - BSIAcr strctUpdateValueACR2300
     - unsigned long unUpdateValueACRType2300
     - unsigned long unUpdateValueKeyIDOrReference2300
     - unsigned long unUpdateValueAuthID2300
     - unsigned long unUpdateValueACRID2300.

2. There exists an unsigned long variable punKeyLen2300.

Test for Assertion 23.1

The function is tested using valid parameters.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGetCryptoProperties().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard2301.

3. (Pre) There exists a target PKI provider module on the connected card with the following properties:
   - the ACR for each of the gscBsiGetChallenge(), gscBsiSkiInternalAuthenticate(), gscBsiPkiCompute(), gscBsiGcDataCreate(), gscBsiGcDataDelete(), gscBsiGcReadTagList(), gscBsiGcReadValue(), and gscBsiGcDataUpdate() services has
     - access method type == BSI_ACR_PIN
     - key ID or reference == _keyIDOrReference1
     - number of access methods logically combined in the ACR == 1
     - ACRID == null
   - the container is represented by AID value ==
     - _goodGSCAID1, of length _goodGSCAID1Len (GSC), or
     - _goodCACAID1, of length _goodCACAID1Len (CAC).

4. (Pre) Print "Testing of Assertion 23.1".

5. Make a gscBsiGetCryptoProperties() call to the SPS, using
   - hCard == hCard2301
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - strctCRYPTOacr == strctCRYPTOacr2300
   - punKeyLen == punKeyLen2300.

Verification Goal:

To verify the Expected Results:

1. The call returns
   - the return code BSI_OK or the return code BSI_NO_CARDSERVICE.

2. If the return code is BSI_OK, then the variables of strctCRYPTOacr2300 are correctly set to indicate access control conditions for all operations.

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiGetCryptoProperties() call returns the code BSI_OK, then:

   Case 1.1: If
   - each of unGetChallengeACRType2300, unInternalAuthenticateACRType2300, unPkiComputeACRType2300, unCreateACRType2300, unDeleteACRType2300, unReadTagListACRType2300, unReadValueACRType2300, and unUpdateValueACRType2300 == BSI_ACR_PIN

   and
   - each of unGetChallengeKeyIDOrReference2300, unInternalAuthenticateKeyIDOrReference2300,

unPkiComputeKeyIDOrReference2300,
unCreateKeyIDOrReference2300,
unDeleteKeyIDOrReference2300,
unReadTagListKeyIDOrReference2300,
unReadValueKeyIDOrReference2300, and
unUpdateKeyIDOrReference2300 == _keyIDOrReference1

and

– each of unGetChallengeAuthNB2300,
unInternalAuthenticateAuthNB2300, unPkiComputeAuthNB2300,
unCreateAuthNB2300, unDeleteAuthNB2300,
unReadTagListAuthNB2300, unReadValueAuthNB2300, and
unUpdateValueAuthNB2300 == 1

and

– each of unGetChallengeACRID2300,
unInternalAuthenticateACRID2300, unPkiComputeACRID2300,
unCreateACRID2300, unDeleteACRID2300,
unReadTagListACRID2300, unReadValueACRID2300, and
unUpdateValueACRID2300 == null

then print
"gscBsiGetCryptoProperties() called with valid parameters has been verified.
Status: Test 23.1 Passed."

Case 1.2: If

– any of unGetChallengeACRType2300,
unInternalAuthenticateACRType2300, unCreateACRType2300,
unPkiComputeACRType2300, unDeleteACRType2300,
unReadTagListACRType2300, unReadValueACRType2300, and
unUpdateValueACRType2300 /= BSI_ACR_PIN

or

– any of unGetChallengeKeyIDOrReference2300,
unInternalAuthenticateKeyIDOrReference2300,
unPkiComputeKeyIDOrReference2300,
unCreateKeyIDOrReference2300,
unDeleteKeyIDOrReference2300,
unReadTagListKeyIDOrReference2300,
unReadValueKeyIDOrReference2300, and
unUpdateKeyIDOrReference2300 /= _keyIDOrReference1

or

– any of unGetChallengeAuthNB2300,
unInternalAuthenticateAuthNB2300, unPkiComputeAuthNB2300,
unCreateAuthNB2300, unDeleteAuthNB2300,
unReadTagListAuthNB2300, unReadValueAuthNB2300, and
unUpdateValueAuthNB2300 /= 1

or

– any of unGetChallengeACRID2300,
unInternalAuthenticateACRID2300, unCreateACRID2300,
unPkiComputeACRID2300, unDeleteACRID2300,
unReadTagListACRID2300, unReadValueACRID2300, and
unUpdateValueACRID2300 /= null

then print
"gscBsiGetCryptoProperties() called with valid parameters has not been verified.
Status: Test 23.1 Failed."

Case 2: If the gscBsiGetCryptoProperties() call returns the code BSI_NO_CARDSERVICE, then print
"gscBsiGetCryptoProperties() is not supported.
Status: Test 23.1 Not Supported."

Case 3: If the gscBsiGetCryptoProperties() call does not return the code BSI_OK or the code BSI_NO_CARDSERVICE, then print
"gscBsiPkiGetCryptoProperties() called with valid parameters returned an incorrect code.
Status: Test 23.1 Failed."

Test for Assertion 23.2

The function is tested using a bad handle.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGetCryptoProperties().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard2302.

3. (Pre) There exists a target PKI provider module on the connected card with the following properties:
   - the ACR for each of the gscBsiGetChallenge(), gscBsiSkiInternalAuthenticate(), gscBsiPkiCompute(), gscBsiGcDataCreate(), gscBsiGcDataDelete(), gscBsiGcReadTagList(), gscBsiGcReadValue(), and gscBsiGcDataUpdate() services has
     - access method type == BSI_ACR_PIN
     - key ID or reference == _keyIDOrReference1
     - number of access methods logically combined in the ACR == 1
     - ACRID == null
   - the container is represented by AID value ==
     - _goodGSCAID1, of length _goodGSCAID1Len (GSC), or
     - _goodCACAID1, of length _goodCACAID1Len (CAC).

6. (Pre) Print "Testing of Assertion 23.2".

7. Make a gscBsiGetCryptoProperties() call to the SPS, using
   - hCard /= hCard2302
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - strctCRYPTOacr == strctCRYPTOacr2300
   - punKeyLen == punKeyLen2300.

Verification Goal:
To verify the Expected Results:

1. The call returns
   - the return code BSI_BAD_HANDLE or the return code BSI_NO_CARDSERVICE.

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiGetCryptoProperties() call returns the code BSI_BAD_HANDLE, then:

   Perform Test for Assertion 9.23.2.1 using
   - hCard == hCard2302.

   Print
   "gscBsiGetCryptoProperties() called with a bad handle has been verified.
   Status: Test 23.2 Passed."

   Case 2: If the gscBsiGetCryptoProperties() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiGetCryptoProperties() is not supported.
   Status: Test 23.2 Not Supported."

   Case 3: If the gscBsiGetCryptoProperties() call does not return the code BSI_BAD_HANDLE or the code BSI_NO_CARDSERVICE, then print
   "gscBsiGetCryptoProperties() called with a bad handle returned an incorrect code.
   Status: Test 23.2 Failed."

Test for Assertion 23.3

The function is tested with another application having established a transaction lock.

Until we encounter implementations that allow multiple simultaneous applications, I don't think we need to worry about this assertion.

Test for Assertion 23.4

The function is tested using a bad AID value.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGetCryptoProperties().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard2304.

3. (Pre) There exists a target PKI provider module on the connected card with the following properties:
   - the ACR for each of the gscBsiGetChallenge(), gscBsiSkiInternalAuthenticate(), gscBsiPkiCompute(), gscBsiGcDataCreate(), gscBsiGcDataDelete(), gscBsiGcReadTagList(), gscBsiGcReadValue(), and gscBsiGcDataUpdate() services has
     - access method type == BSI_ACR_PIN
     - key ID or reference == _keyIDOrReference1
     - number of access methods logically combined in the ACR == 1
     - ACRID == null

- the container is represented by AID value ==
  - _goodGSCAID1, of length _goodGSCAID1Len (GSC), or
  - _goodCACAID1, of length _goodCACAID1Len (CAC).

4. (Pre) There does not exist a container on the connected card with AID value ==
   - _badGSCAID, of length _badGSCAIDLen (GSC), or
   - _badCACAID, of length _badCACAIDLen (CAC).

5. (Pre) Print "Testing of Assertion 23.4".

6. Make a gscBsiGetCryptoProperties() call to the SPS, using
   - hCard == hCard2304
   - uszAID == _badGSCAID (GSC) or _badCACAID (CAC)
   - unAIDLen == _badGSCAIDLen (GSC) or _badCACAIDLen (CAC)
   - strctCRYPTOacr == strctCRYPTOacr2300
   - punKeyLen == punKeyLen2300.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_BAD_AID or the return code BSI_NO_CARDSERVICE.

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiGetCryptoProperties() call returns the code BSI_BAD_AID, then:

   Perform Test for Assertion 9.23.4.1 using
   - hCard == hCard2302.

   Print
   "gscBsiGetCryptoProperties() called with a bad AID value has been verified.
   Status: Test 23.4 Passed."

   Case 2: If the gscBsiGetCryptoProperties() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiGetCryptoProperties() is not supported.
   Status: Test 23.4 Not Supported."

   Case 3: If the gscBsiGetCryptoProperties() call does not return the code BSI_BAD_AID or the code BSI_NO_CARDSERVICE, then print
   "gscBsiGetCryptoProperties() called with a bad AID value returned an incorrect code.
   Status: Test 23.4 Failed."

Test for Assertion 23.5

The function is tested using a bad AID length.

Instantiation Scenario:
1. (Pre) Construct the Starting State for the testing of gscBsiGetCryptoProperties().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard2305.

3. (Pre) There exists a target PKI provider module on the connected card with the following properties:
   - the ACR for each of the gscBsiGetChallenge(), gscBsiSkiInternalAuthenticate(), gscBsiPkiCompute(), gscBsiGcDataCreate(), gscBsiGcDataDelete(), gscBsiGcReadTagList(), gscBsiGcReadValue(), and gscBsiGcDataUpdate() services has
     - access method type == BSI_ACR_PIN
     - key ID or reference == _keyIDOrReference1
     - number of access methods logically combined in the ACR == 1
     - ACRID == null
   - the container is represented by AID value ==
     - _goodGSCAID1, of length _goodGSCAID1Len (GSC), or
     - _goodCACAID1, of length _goodCACAID1Len (CAC).

4. (Pre) Print "Testing of Assertion 23.5".

5. Make a gscBsiGetCryptoProperties() call to the SPS, using
   - hCard == hCard2305
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _badAIDLen
   - strctCRYPTOacr == strctCRYPTOacr2300
   - punKeyLen == punKeyLen2300.

Verification Goal:
To verify the Expected Results:
1. The call returns
   - the return code BSI_BAD_PARAM or the return code BSI_NO_CARDSERVICE.

Verification and Reporting Scenario:
1. **Case 1**: If the gscBsiGetCryptoProperties() call returns the code BSI_BAD_PARAM, then:

   Perform Test for Assertion 9.23.5.1 using
   - hCard == hCard2305.

   Print
   "gscBsiGetCryptoProperties() called with a bad AID length has been verified.
   Status: Test 23.5 Passed."

   Case 2: If the gscBsiGetCryptoProperties() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiGetCryptoProperties() is not supported.
   Status: Test 23.5 Not Supported."

   Case 3: If the gscBsiGetCryptoProperties() call does not return the code BSI_BAD_PARAM or the code BSI_NO_CARDSERVICE, then print
   "gscBsiGetCryptoProperties() called with a bad AID length returned an incorrect code.
   Status: Test 23.5 Failed."

Test for Assertion 23.6

The function is tested with a removed card.

Instantiation Scenario:

1. (Pre) Construct the Starting State for the testing of gscBsiGetCryptoProperties().

2. (Pre) A card that claims conformance to the GSC-IS is in a reader, connected with handle hCard2306.

3. (Pre) There exists a target PKI provider module on the connected card with the following properties:
   - the ACR for each of the gscBsiGetChallenge(), gscBsiSkiInternalAuthenticate(), gscBsiPkiCompute(), gscBsiGcDataCreate(), gscBsiGcDataDelete(), gscBsiGcReadTagList(), gscBsiGcReadValue(), and gscBsiGcDataUpdate() services has
     - access method type == BSI_ACR_PIN
     - key ID or reference == _keyIDOrReference1
     - number of access methods logically combined in the ACR == 1
     - ACRID == null
   - the container is represented by AID value ==
     - _goodGSCAID1, of length _goodGSCAID1Len (GSC), or
     - _goodCACAID1, of length _goodCACAID1Len (CAC).

4. (Pre) Remove the connected card from the reader.

5. (Pre) Print "Testing of Assertion 23.6".

6. Make a gscBsiGetCryptoProperties() call to the SPS, using
   - hCard == hCard2306
   - uszAID == _goodGSCAID1 (GSC) or _goodCACAID1 (CAC)
   - unAIDLen == _goodGSCAID1Len (GSC) or _goodCACAID1Len (CAC)
   - strctCRYPTOacr == strctCRYPTOacr2300
   - punKeyLen == punKeyLen2300.

Verification Goal:

To verify the Expected Results:

1. The call returns
   - the return code BSI_CARD_REMOVED or the return code BSI_NO_CARDSERVICE.

Verification and Reporting Scenario:

1. **Case 1**: If the gscBsiGetCryptoProperties() call returns the code BSI_CARD_REMOVED, then print
   "gscBsiGetCryptoProperties() called with the connected card removed has been verified.
   Status: Test 23.6 Passed."

   Case 2: If the gscBsiGetCryptoProperties() call returns the code BSI_NO_CARDSERVICE, then print
   "gscBsiGetCryptoProperties() is not supported.

Status: Test 23.6 Not Supported."

Case 3: If the gscBsiGetCryptoProperties() call does not return the code BSI_CARD_REMOVED or the code BSI_NO_CARDSERVICE, then print

"gscBsiGetCryptoProperties() called with the connected card removed returned an incorrect code.
Status: Test 23.6 Failed."

Appendix A

List of Symbolic Constants

_badAIDLen == 0

_badAuthValue == {'B','A','D',' ','A','U','T','H'}

_badAuthValueLen == 8

_badCACAID == "A0000000793333"

_badCACAIDLen == 14

_badCardCommand == {1}

_badChallenge == null

_badChallengeLen == 0

_badDvalueLen == 0

_badGSCAID == "A0000001163333"

_badGSCAIDLen == 14

_badMessage == null

_badMessageLen == 0

_badReaderName == "Bad Reader Name"

_badReaderNameLen == 15

_existingDvalueFull == $\{F_{16},F_{16},F_{16},F_{16},F_{16},F_{16},F_{16},F_{16},F_{16},F_{16}\}$

_existingDvalueFullLen == 10

_existingDvalueLen == 4

_existingDvalue1309 == {0,0,0,1}

_existingDvalue1310 == {0,0,0,2}

_existingDvalue1401 == {0,0,0,3}

_existingDvalue1402 == {0,0,0,4}

_existingDvalue1403 == {0,0,0,5}

_existingDvalue1404 == {0,0,0,6}

_existingDvalue1405 == {0,0,0,7}

_existingDvalue1406 == {0,0,0,8}

_existingDvalue1407 == {0,0,0,9}

_existingDvalue1408 == {0,0,0,A_{16}}

_existingDvalue1701 == {0,0,0,B_{16}}

_existingDvalue1702 == {0,0,0,C_{16}}

_existingDvalue1703 == {0,0,0,D_{16}}

_existingDvalue1704 == {0,0,0,E_{16}}

_existingDvalue1706 == {0,0,0,F_{16}}

_existingDvalue1707 == {0,0,0,G_{16}}

_existingDvalue1709 == {0,0,0,H_{16}}

_existingDvalue1710 == {0,0,1,0}

_existingDvalue1801 == {0,0,1,1}

_existingDvalue1802 == {0,0,1,2}

_existingDvalue1803 == {0,0,1,3}

_existingDvalue1804 == {0,0,1,4}

_existingDvalue1805 == {0,0,1,5}

_existingDvalue1806 == {0,0,1,6}

_existingDvalue1807 == {0,0,1,7}

_existingDvalue1808 == {0,0,1,8}

_existingDvalue1809 == {0,0,1,9}

_existingTagFull == 1

_existingTag1309 == 2

_existingTag1310 == 3

_existingTag1401 == 4

_existingTag1402 == 5

_existingTag1403 == 6

_existingTag1404 == 7

_existingTag1405 == 8

_existingTag1406 == 9

_existingTag1407 == 10

_existingTag1408 == 11

_existingTag1701 == 12

_existingTag1702 == 13

_existingTag1703 == 14

_existingTag1704 == 15

_existingTag1706 == 16

_existingTag1707 == 17

_existingTag1709 == 18

_existingTag1710 == 19

_existingTag1801 == 20

_existingTag1802 == 21

_existingTag1803 == 22

_existingTag1804 == 23

_existingTag1805 == 24

_existingTag1806 == 25

_existingTag1807 == 26

_existingTag1808 == 27

_existingTag1809 == 28

_goodAlgoID1 == DES3-ECB == 81_{16}

_goodAlgoID2 == RSA_NO_PAD == $A3_{16}$

_goodAuthValue1 == {'1','2','3','4','5','6','7','8'}

_goodAuthValue1Len == 8

_goodAuthValue2 == {'8','7','6','5','4','3','2','1'}

_goodAuthValue2Len == 8

_goodCACAID1 == "A0000000792222"

_goodCACAID1Len == 14

_goodCACAID2 == "A0000000791111"

_goodCACAID2Len == 14

_goodCardCommand == {0,0,A_{16},4,0,3,0,0,0,2,3,F_{16},0,0}

_goodCardCommandLen == 14

_goodCardResponse == {{6,2,8,3}, {6,2,8,4}, {6,A_{16},8,1}, {6,A_{16},8,2},
{6,A_{16},8,6}, {6,A_{16},8,7}, {9,0,0,0}}

_goodChallenge == {'g','o','o','d',
' ','c','h','a','l','l','e','n','g','e'}

_goodChallengeLen == 14

_goodGSCAID1 == "A0000001162222"

_goodGSCAID1Len == 14

_goodGSCAID2 == "A0000001161111"

_goodGSCAID2Len == 14

_goodMessage == {'g','o','o','d',' ','m','e','s','s','a','g','e'}

_goodMessageLen == 12

_invalidAlgoID == -1

_invalidTag == 0

_keyIDOrReference1 == 1

_keyIDOrReference2 == 1

_newDvalueLen == 4

_newDvalue1301 == {1,0,0,1}

_newDvalue1302 == {1,0,0,2}

_newDvalue1303 == {1,0,0,3}

_newDvalue1304 == {1,0,0,4}

_newDvalue1305 == {1,0,0,5}

_newDvalue1306 == {1,0,0,6}

_newDvalue1307 == {1,0,0,7}

_newDvalue1308 == {1,0,0,8}

_newDvalue1309 == {1,0,0,9}

_newDvalue1310 == {1,0,0,A_{16}}

_newDvalue1801 == {1,0,0,B_{16}}

_newDvalue1802 == {1,0,0,C_{16}}

_newDvalue1803 == {1,0,0,D_{16}}

_newDvalue1804 == {1,0,0,E_{16}}

_newDvalue1805 == {1,0,0,F_{16}}

_newDvalue1806 == {1,0,1,0}

_newDvalue1807 == {1,0,1,1}

_newDvalue1808 == {1,0,1,2}

_newDvalue1809 == {1,0,1,3}

_newTag1301 == 101

_newTag1302 == 102

_newTag1303 == 103

_newTag1305 == 104

_newTag1306 == 105

_newTag1307 == 106

_newTag1308 == 107

_newTag1309 == 108

_newTag1404 == 109

_newTag1406 == 110

_newTag1705 == 111

_newTag1708 == 112

_newTag1806 == 113

_newTag1807 == 114

_PIN == {'1','2','3','4','5','6','7','8'}

_tooBigDvalue == {1,2,3,4,5,6,7,8,9,A_{16},B_{16},C_{16},D_{16},E_{16},F_{16}}

_tooBigDvalueLen == 15

Note: Some of the above symbolic constants are not used in the C test scenarios. They are used in the Java test scenarios, and are included here to completely define the specification, in Appendix B, of cards required for BSI testing.

Appendix B

List of Cards Required for Testing of BSI Implementations

Card 1. Can be used for tests 1-all, 2.1, 2.2, 2.3, 2.4, 2.5(C binding), 3-all, 4-all, 5-all, 7-all, 8-all, 11-all, 12-all, 13-all, 14-all, 15-all, 16-all, 17-all, 18-all, 19-all:

- claims conformance to the GSC-IS
- has a container for which
  - createACR, deleteACR, readTagListACR, readValueACR, and updateValueACR are PIN Protected
  - PIN == _PIN
  - key ID or reference == _keyIDOrReference1
  - number of access methods logically combined in the ACR == 1
  - ACRID == null
  - AID == _goodCACAID1 (CAC) or _goodGSCAID1 (GSC)
  - can accommodate a new data item of length _newDvalueLen
  - there exist the following data items (TLV):
    - o {_existingTag1309, 4, _existingDvalue1309}
    - o {_existingTag1310, 4, _existingDvalue1310}
    - o {_existingTag1401, 4, _existingDvalue1401}
    - o {_existingTag1402, 4, _existingDvalue1402}
    - o {_existingTag1403, 4, _existingDvalue1403}
    - o {_existingTag1404, 4, _existingDvalue1404}
    - o {_existingTag1405, 4, _existingDvalue1405}
    - o {_existingTag1406, 4, _existingDvalue1406}
    - o {_existingTag1407, 4, _existingDvalue1407}
    - o {_existingTag1408, 4, _existingDvalue1408}
    - o {_existingTag1701, 4, _existingDvalue1701}
    - o {_existingTag1702, 4, _existingDvalue1702}
    - o {_existingTag1703, 4, _existingDvalue1703}
    - o {_existingTag1704, 4, _existingDvalue1704}
    - o {_existingTag1706, 4, _existingDvalue1706}
    - o {_existingTag1707, 4, _existingDvalue1707}
    - o {_existingTag1709, 4, _existingDvalue1709}
    - o {_existingTag1710, 4, _existingDvalue1710}
    - o {_existingTag1801, 4, _existingDvalue1801}
    - o {_existingTag1802, 4, _existingDvalue1802}
    - o {_existingTag1803, 4, _existingDvalue1803}
    - o {_existingTag1804, 4, _existingDvalue1804}
    - o {_existingTag1805, 4, _existingDvalue1805}
    - o {_existingTag1806, 4, _existingDvalue1806}
    - o {_existingTag1807, 4, _existingDvalue1807}
    - o {_existingTag1808, 4, _existingDvalue1808)
    - o {_existingTag1809, 4, _existingDvalue1809}
- has a second container for which
  - updateValueACR is PIN Protected
  - PIN == _PIN
  - AID == _goodCACAID2 (CAC) or _goodGSCAID2 (GSC)
  - there exists one data item:
    - o {_existingTagFull, 10, _existingDvalueFull}

    which comprises the entire container
- neither container has a data item for which tag ==

    - _newTag1301
    - _newTag1302
    - _newTag1303
    - _newTag1305
    - _newTag1306
    - _newTag1307
    - _newTag1308
    - _newTag1309
    - _newTag1404
    - _newTag1406
    - _newTag1705
    - _newTag1708
    - _newTag1806
    - _newTag1807
  - does not have a container for which
    - AID == _badCACAID (CAC) or _badGSCAID (GSC)

Card 2. Can be used for test 2.5 (Java binding), 2.6 (C bonding):
- bad card

Card 3. Can be used for tests 20-all, 21-all, 22-all, 23-all:
- claims conformance to the GSC-IS
- has a PKI provider container for which
  - dataCreateACR, dataDeleteACR, readTagListACR, readValueACR, dataUpdateACR, pkiComputeACR, internalAuthenticateACR, and getChallengeACR are PIN Protected
  - PIN == _PIN
  - AID == _goodCACAID1 (CAC) or _goodGSCAID1 (GSC)
- has an SKI provider container for which
  - pkiComputeACR, internalAuthenticateACR, and readValueACR are PIN Protected
  - PIN == _PIN
  - AID == _goodCACAID2 (CAC) or _goodGSCAID2 (GSC)
- does not have a container for which
  - AID == _badCACAID (CAC) or _badGSCAID (GSC)