# Services Runtime Services Test

## Variable Services Test

Reference Document:

*UEFI Specification*, Variable Services Section.

* Variable Services Functions

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| GetVariable() | Runtime | Returns the value of a variable. |
| GetNextVariableName() | Runtime | Enumerates the current variable names. |
| SetVariable() | Runtime | Sets the value of a variable. |
| QueryVariableInfo() | Runtime | Queries the information about the variables. |

* GetVariable()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.2.1.1.1 | 0xb0d54fee, 0x2787, 0x4d2d, 0xbf, 0x98, 0x73, 0xa0, 0xcd, 0x7f, 0xe9, 0x5d | RT.GetVariable – GetVariable() returns EFI\_INVALID\_PARAMETER with a **VariableName** value of NULL. | 1. Call GetVariable() service with a **VariableName** value of NULL. The return code must be EFI\_INVALID\_PARAMETER. |
| 5.2.1.1.2 | 0x390c5e26, 0x9b46, 0x4974, 0xb3, 0x2d, 0x2b, 0xb1, 0xd4, 0x05, 0xb0, 0xd7 | RT.GetVariable – GetVariable() returns EFI\_INVALID\_PARAMETER with a **VendorGuid** value of NULL. | 1. Call GetVariable() service with a **VendorGuid** value of NULL. The return code must be EFI\_INVALID\_PARAMETER. |
| 5.2.1.1.3 | 0x176354a6, 0x1088, 0x474f, 0xbf, 0x6f, 0x95, 0x8c, 0x1c, 0xc3, 0x40, 0x8f | RT.GetVariable – GetVariable() returns EFI\_INVALID\_PARAMETER with a **DataSize** value of NULL. | 1. Call SetVariable() service to insert a test variable.  2. Call GetVariable() service to get the test variable while the **DataSize** is NULL. The return code must be EFI\_INVALID\_PARAMETER. |
| 5.2.1.1.4 | 0x400ab801, 0xf6c6, 0x4d04, 0xa0, 0x42, 0xa2, 0x15, 0x0b, 0xd5, 0xb6, 0x2a | RT.GetVariable – GetVariable() returns EFI\_INVALID\_PARAMETER with a **Data** value of NULL. | 1. Call SetVariable() service to insert a test variable.  2. Call GetVariable() service to get the test variable while the **Data** is NULL. The return code must be EFI\_INVALID\_PARAMETER. |
| 5.2.1.1.5 | 0x9b704b3d, 0x05a4, 0x4147, 0xb2, 0x55, 0x35, 0xbc, 0x3d, 0xd6, 0xcc, 0x24 | RT.GetVariable – GetVariable() returns EFI\_NOT\_FOUND with a nonexistent variable. | 1. Call SetVariable() service to insert a test variable.  2. Call SetVariable() service to delete the test variable.  3. Call GetVariable() service to get the test variable. The return code must be EFI\_NOT\_FOUND.  4. Call SetVariable() services to insert two variables that are similar to the test variable.  5. Call GetVariable() service to get the test variable. The return code must be EFI\_NOT\_FOUND. |
| 5.2.1.1.6 | 0xd3d915a5, 0xe7b0, 0x4417, 0x9c, 0x2e, 0x1a, 0xa8, 0x42, 0x4d, 0x22, 0x2c | RT.GetVariable – GetVariable() returns EFI\_NOT\_FOUND with a nonexistent **VendorGuid**. | 1. Call SetVariable() service to insert a test variable with GUID2.  2. Call GetVariable() service to get the variable with GUID1. The return code must be EFI\_NOT\_FOUND. |
| 5.2.1.1.7 | 0x1562ce35, 0x83e7, 0x48a7, 0xad, 0x71, 0xfa, 0xa4, 0xbe, 0x17, 0x88, 0x46 | RT.GetVariable – GetVariable() returns EFI\_BUFFER\_TOO\_SMALL with a **DataSize** value of 0. | 1. Call SetVariable() service to insert a test variable.  2. Call GetVariable() service to get the test variable with a **DataSize** value of 0. The return code must be EFI\_BUFFER\_TOO\_SMALL, and the returned **DataSize** should be the inserted value in step 1. |
| 5.2.1.1.8 | 0x121c17d1, 0xbb0e, 0x4e2e, 0xb2, 0xa5, 0x03, 0x86, 0x2f, 0x46, 0xc0, 0x39 | RT.GetVariable – GetVariable() returns EFI\_BUFFER\_TOO\_SMALL with a **DataSize** value of -1. | 1. Call SetVariable() service to insert a test variable.  2. Call GetVariable() service to get the test variable with the inserted **DataSize** value of –1. The return code must be EFI\_BUFFER\_TOO\_SMALL, and the returned **DataSize** should be the inserted value in step 1. |
| 5.2.1.1.9 | 0xe542e81c, 0x2020, 0x4f3e, 0xa9, 0xb, 0x67, 0xd4, 0xa8, 0xd1, 0x70, 0xb4 | RT.GetVariable – GetVariable() returns EFI\_BUFFER\_TOO\_SMALL with a *DataSize* value of 0. | 1. Call SetVariable() service to insert a test variable. 2. Call GetVariable() service to get the test variable with a *DataSize* value of 0 and NULL *Data*. The return code must be EFI\_BUFFER\_TOO\_SMALL, and the returned *DataSize* should be the inserted value in step 1. |
| 5.2.1.1.10 | 0xaa35cc00, 0xc55c, 0x42d8, 0xa6, 0xd4, 0x1e, 0xb4, 0x9d, 0xe3, 0xd7, 0x54 | RT.GetVariable – GetVariable() gets the existing variable without attributes at EFI\_TPL\_APPLICATION. | 1. Call SetVariable() service to insert a test variable. 2. Call GetVariable() service to get the test variable without *Attributes*. The returned status must be EFI\_SUCCESS, and the returned *Data* and *DataSize* must be the same as the data written before. |
| 5.2.1.1.11 | 0x742a9651, 0x9783, 0x43b8, 0x8c, 0x18, 0x47, 0x04, 0xae, 0x41, 0xc3, 0x34 | RT.GetVariable – GetVariable() gets the existing variable without attributes at EFI\_TPL\_CALLBACK. | 1. Call SetVariable() service to insert a test variable. 2. Call GetVariable() service to get the test variable without *Attributes*. The returned status must be EFI\_SUCCESS, and the returned *Data* and *DataSize* must be the same as the data written before. |
| 5.2.1.1.12 | 0x90e959d0, 0xbe2c, 0x45fd, 0x85, 0x32, 0x85, 0x21, 0xe4, 0xe0, 0xfb, 0x72 | RT.GetVariable – GetVariable() gets the existing variable with attributes at EFI\_TPL\_APPLICATION | 1. Call SetVariable() service to insert a test variable. 2. Call GetVariable() service to get the test variable with *Attributes*. The returned status must be EFI\_SUCCESS, and the returned *Attributes*, *Data* and *DataSize* must be the same as the data written before. |
| 5.2.1.1.13 | 0x5c8b43b7, 0xec6f, 0x4621, 0xb8, 0x48, 0x6a, 0x40, 0x0f, 0xd8, 0xb3, 0x43 | RT.GetVariable – GetVariable() gets the existing variable with attributes at EFI\_TPL\_CALLBACK. | 1. Call SetVariable() service to insert a test variable. 2. Call GetVariable() service to get the test variable with *Attributes*. The returned status must be EFI\_SUCCESS, and the returned *Attributes*, *Data* and *DataSize* must be the same as the data written before. |

### GetVariable()

### GetNextVariableName()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.2.1.2.1 | 0x5826847a, 0x9067, 0x4f9f, 0x88, 0x38, 0x0b, 0xf8, 0xec, 0x20, 0x17, 0x1c | RT.GetNextVariableName – GetNextVariableName() returns EFI\_INVALID\_PARAMETER with a **VariableNameSize** value of NULL. | 1. Call GetNextVariableName() service with a **VariableNameSize** value of NULL. The return code must be EFI\_INVALID\_PARAMETER. |
| 5.2.1.2.2 | 0x8e8258dc, 0x6634, 0x4de1, 0x85, 0x7a, 0x60, 0x45, 0x7e, 0xfa, 0x7c, 0x21 | RT.GetNextVariableName - GetNextVariableName() returns EFI\_INVALID\_PARAMETER with a **VariableName** value of NULL. | 1. Call GetNextVariableName() service with a **VariableName** value of NULL. The returned status should be EFI\_INVALID\_PARAMETER. |
| 5.2.1.2.3 | 0x99a357f0, 0xb6c5, 0x4aec, 0x96, 0x48, 0x34, 0x73, 0x2d, 0x2a, 0x49, 0x50 | RT.GetNextVariableName - GetNextVariableName() returns EFI\_INVALID\_PARAMETER with a **VendorGuid** value of NULL. | 1. Call GetNextVariableName() service with a **VendorGuid** value of NULL. The returned status should be EFI\_INVALID\_PARAMETER. |
| 5.2.1.2.4 | 0x51c19dba, 0xbaf6, 0x4854, 0xac, 0x09, 0x60, 0x45, 0x47, 0x88, 0x67, 0x98 | RT.GetNextVariableName – GetNextVariableName() returns EFI\_BUFFER\_TOO\_SMALL with a **VariableNameSize** value of 2. | 1. Call SetVariable() service to insert a test variable.  2. Call GetNextVariableName() service with a **VariableNameSize** value of 2. The returned status should be EFI\_BUFFER\_TOO\_SMALL. |
| 5.2.1.2.5 | 0xfe09ff82, 0xb289, 0x449f, 0xb0, 0x83, 0x98, 0x1d, 0x68, 0xd9, 0x17, 0xb1 | RT.GetNextVariableName – GetNextVariableName() returns EFI\_NOT\_FOUND after the entire variable list returned. | 1. Call SetVariable() service to insert a test variable.  2. Call SetVariable() service to delete the test variable.  3. Call GetNextVariableName() service to traverseall variables. The deleted test variable should not be returned.  4. The last returned status of GetNextVariableName() service should be EFI\_NOT\_FOUND. |
| 5.2.1.2.6 | 0x12071508, 0x16c7, 0x4e5e, 0xa4, 0x22, 0x59, 0xe0, 0x24, 0x1c, 0xc6, 0x28 | RT.GetNextVariableName – GetNextVariableName() gets the existing variable at EFI\_TPL\_APPLICATION. | 1. Call SetVariable() service to insert a test variable.  2. Call GetNextVariableName() service to traverseall variables. The test variable should be returned in this loop. |
| 5.2.1.2.7 | 0xa85043bc, 0x4f0d, 0x47b3, 0x8e, 0x9d, 0x2d, 0xb6, 0xc8, 0xf8, 0xfa, 0xef | RT.GetNextVariableName – GetNextVariableName() gets the exist variable at EFI\_TPL\_CALLBACK. | 1. Call SetVariable() service to insert a test variable.  2. Call GetNextVariableName() service to traverseall variables. The test variable should be returned in this loop. |

### SetVariable()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.2.1.3.1 | 0x73af529b, 0x3ebe, 0x464a, 0xba, 0x6a, 0xfb, 0x04, 0x7b, 0x56, 0x4f, 0x74 | RT.SetVariable – SetVariable() returns EFI\_INVALID\_PARAMETER when the **VariableName** value is an empty string. | 1. Call SetVariable() service when the **VariableName** value is an empty string. The returned status should be EFI\_INVALID\_PARAMETER. |
| 5.2.1.3.2 | 0x39e95cbb, 0x6b89, 0x473e, 0x91, 0xba, 0x92, 0x08, 0x2d, 0x1b, 0x94, 0xad | RT.SetVariable – SetVariable() returns EFI\_INVALID\_PARAMETER with RA only **Attributes**. | 1. Call SetVariable() service with EFI\_VARIABLE\_RUNTIME\_ACCESS attributes. The returned status should be EFI\_INVALID\_PARAMETER. |
| 5.2.1.3.3 | 0xf6ef5087, 0x4962, 0x4d71, 0x80, 0x09, 0xdb, 0xe2, 0x78, 0x94, 0x53, 0xe6 | RT.SetVariable – SetVariable() returns EFI\_INVALID\_PARAMETER with NV|RA **Attributes**. | 1. Call SetVariable() service with EFI\_VARIABLE\_NON\_VOLATILE | EFI\_VARIABLE\_RUNTIME\_ACCESS attributes. The returned status should be EFI\_INVALID\_PARAMETER. |
| 5.2.1.3.4 | 0x65973462, 0x6877, 0x408f, 0x9b, 0xe1, 0x46, 0x69, 0x3e, 0xab, 0x03, 0x84 | RT.SetVariable – SetVariable() returns EFI\_INVALID\_PARAMETER with a variable that exceeds the maximum size. | 1. Call SetVariable() service to set a test variable with the size of (UINTN)-1. The returned status should be EFI\_INVALID\_PARAMETER. |
| 5.2.1.3.5 | 0x6c9cf2ea, 0xcabd, 0x4312, 0xb9, 0xcf, 0x0a, 0x96, 0xc4, 0xf1, 0xea, 0x8b | RT.SetVariable – SetVariable() sets a nonexistent variable at EFI\_TPL\_APPLICATION. | 1. Call SetVariable() service to insert a test variable with GUID2.  2. Call SetVariable() service to insert a test variable with GUID1. The returned status must be EFI\_SUCCESS.  3. Call GetVariable() service to get the test variable with GUID1 and GUID2. The data of both variables should be the same as the values written before. |
| 5.2.1.3.6 | 0x3ae09eaf, 0x07cd, 0x4320, 0x92, 0xfd, 0xe9, 0xe6, 0x4b, 0x31, 0x6f, 0xe1 | RT.SetVariable – SetVariable() sets a nonexistent variable at EFI\_TPL\_CALLBACK. | 1. Call SetVariable() service to insert a test variable with GUID2.  2. Call SetVariable() service to insert a test variable with GUID1. The returned status must be EFI\_SUCCESS.  3. Call GetVariable() service to get the test variable with GUID1 and GUID2. The data of both variables should be the same as the values written before. |
| 5.2.1.3.7 | 0x7ccde75b, 0x4ef2, 0x40ec, 0x9a, 0xcb, 0x84, 0x7b, 0xb5, 0x29, 0x73, 0xbe | RT.SetVariable – SetVariable() sets the existing variable with the data from EFI\_TPL\_APPLICATION. | 1. Call SetVariable() service to insert a test variable.  2. Call SetVariable() service to insert the test variable again with the same data. The returned status should be EFI\_SUCCESS.  3. Call GetVariable() service to get the test variable. The data of the test variable should be unchanged. |
| 5.2.1.3.8 | 0x5b720ad1, 0xd0cc, 0x4be0, 0x93, 0x18, 0x20, 0x1b, 0xac, 0x32, 0x8d, 0x4f | RT.SetVariable – SetVariable() sets the existing variable with the data from EFI\_TPL\_CALLBACK. | 1. Call SetVariable() service to insert a test variable.  2. Call SetVariable() service to insert the test variable again with the same data. The returned status should be EFI\_SUCCESS.  3. Call GetVariable() service to get the test variable. The data of the test variable should be unchanged. |
| 5.2.1.3.9 | 0x2dee62d3, 0xbab7, 0x4d91, 0x8b, 0x47, 0x3e, 0x38, 0x35, 0xd3, 0x88, 0xae | RT.SetVariable – SetVariable() sets the existing variable value which is different from the one at EFI\_TPL\_APPLICATION. | 1. Call SetVariable() service to insert a test variable.  2. Call SetVariable() service to insert the test variable again with the different data in which the left part of new data is the same as old data. The returned status should be EFI\_SUCCESS.  3. Call GetVariable() service to get the test variable. The data of the test variable should be changed to the new one. |
| 5.2.1.3.10 | 0x861a0691, 0x6590, 0x4a28, 0xae, 0x56, 0xaa, 0xcb, 0xf3, 0xf2, 0xbe, 0x99 | RT.SetVariable – SetVariable() sets the existing variable value which is different from the one at EFI\_TPL\_CALLBACK. | 1. Call SetVariable() service to insert a test variable.  2. Call SetVariable() service to insert the test variable again with the different data in which the left part of new data is the same as the old data The returned status should be EFI\_SUCCESS.  3. Call GetVariable() service to get the test variable. The data of the test variable should be changed to the new one. |
| 5.2.1.3.11 | 0x76198a1a, 0xc63a, 0x4a3b, 0x88, 0xb0, 0xc4, 0x45, 0x39, 0xdd, 0xff, 0x5d | RT.SetVariable – SetVariable() sets the existing variable with different data at EFI\_TPL\_APPLICATION. | 1. Call SetVariable() service to insert a test variable.  2. Call SetVariable() service to insert the test variable again with the different data in which the left part of old data is the same as the new data The returned status should be EFI\_SUCCESS.  3. Call GetVariable() service to get the test variable. The data of the test variable should be changed to the new one. |
| 5.2.1.3.12 | 0xcefbdb2c, 0x0c7d, 0x4dcf, 0xae, 0x16, 0x32, 0xa8, 0x78, 0xca, 0x2d, 0x3e | RT.SetVariable – SetVariable() sets the existing variable with different data at EFI\_TPL\_CALLBACK. | 1. Call SetVariable() service to insert a test variable.  2. Call SetVariable() service to insert the test variable again with the different data in which the left part of old data is the same as the new data. The returned status should be EFI\_SUCCESS.  3. Call GetVariable() service to get the test variable. The data of the test variable should be changed to the new one. |
| 5.2.1.3.13 | 0xc457149c, 0x75d0, 0x48b5, 0xa1, 0x6c, 0x7e, 0x9f, 0x14, 0x4a, 0xab, 0x15 | RT.SetVariable – SetVariable() sets similar existing variables at EFI\_TPL\_APPLICATION. | 1. Call SetVariable() service to insert two similar variables.  2. Call SetVariable() service to insert a test variable. The returned status should be EFI\_SUCCESS.  3. Call GetVariable() service to get the test variable. The returned data should be those written before. |
| 5.2.1.3.14 | 0x89f533da, 0x20ee, 0x41f8, 0x8c, 0x60, 0xc3, 0xc4, 0x14, 0x19, 0x05, 0x15 | RT.SetVariable – SetVariable() sets similar existing variables at EFI\_TPL\_CALLBACK. | 1. Call SetVariable() service to insert two similar variables.  2. Call SetVariable() service to insert a test variable. The returned status should be EFI\_SUCCESS.  3. Call GetVariable() service to get the test variable. The returned data should be those written before. |
| 5.2.1.3.15 | 0xfc5f89d1, 0x4fce, 0x4fe9, 0xa2, 0xfd, 0xa2, 0xfe, 0x69, 0x5b, 0xaa, 0x35 | RT.SetVariable – SetVariable() sets similar existing variables at EFI\_TPL\_APPLICATION. | 1. Call SetVariable() service to insert a similar variable, whose name is the test variable’s name plus character ‘A’.  2. Call SetVariable() service to insert a test variable. The returned status should be EFI\_SUCCESS.  3. Call GetVariable() service to get the similar variable. The returned data should be unchanged. |
| 5.2.1.3.16 | 0xfa5f4961, 0xdfaf, 0x425f, 0x95, 0x14, 0x14, 0x52, 0x5c, 0x69, 0xc7, 0x83 | RT.SetVariable – SetVariable() sets similar existing variables at EFI\_TPL\_CALLBACK. | 1. Call SetVariable() service to insert a similar variable, whose name is the test variable’s name + ‘A’.  2. Call SetVariable() service to insert a test variable. The returned status should be EFI\_SUCCESS.  3. Call GetVariable() service to get the similar variable. The returned data should be unchanged. |
| 5.2.1.3.17 | 0x3cf290ca, 0x49e9, 0x43c0, 0x8a, 0x0c, 0x46, 0xea, 0x17, 0x53, 0x41, 0x08 | RT.SetVariable – SetVariable() sets similar existing variables at EFI\_TPL\_APPLICATION. | 1. Call SetVariable() service to insert a similar variable, whose name is the test variable’s name minus character ‘A’.  2. Call SetVariable() service to insert a test variable. The returned status should be EFI\_SUCCESS.  3. Call GetVariable() service to get a similar variable. The returned data should be unchanged. |
| 5.2.1.3.18 | 0xc1f69f8f,  0xa6ed,  0x4823, 0x88,  0xd9, 0x9a,  0x23, 0x8e,  0x6a, 0x11,  0x00 | RT.SetVariable – SetVariable() sets similar existing variables at EFI\_TPL\_CALLBACK. | 1. Call SetVariable() service to insert a similar variable, whose name is the test variable’s name minus character ‘A’.  2. Call SetVariable() service to insert a test variable. The returned status should be EFI\_SUCCESS.  3. Call GetVariable() service to get the similar variable. The returned data should be unchanged. |
| 5.2.1.3.19 | 0x7b893a77, 0x70ca, 0x48e4, 0xad, 0x1d, 0xe4, 0x31, 0x15, 0xb1, 0xce, 0x5e | RT.SetVariable – SetVariable() removes all variables with a **DataSize** value of 0 at EFI\_TPL\_APPLICATION. | 1. Call SetVariable() service to insert a test variable.  2. Call SetVariable() service with a **DataSize** value of 0 to delete the test variable. The returned status should be EFI\_SUCCESS.  3. Call GetVariable() service to get the test variable. The returned status should be EFI\_NOT\_FOUND. |
| 5.2.1.3.20 | 0x8fcc7182, 0x4f77, 0x4841, 0xbb, 0x81, 0x20, 0xe5, 0x30, 0x5e, 0xa9, 0xda | RT.SetVariable – SetVariable() removes all variables with a **DataSize** value of 0 at EFI\_TPL\_CALLBACK. | 1. Call SetVariable() service to insert a test variable.  2. Call SetVariable() service with a **DataSize** value of 0 to delete the test variable. The returned status should be EFI\_SUCCESS.  3. Call GetVariable() service to get the test variable. The returned status should be EFI\_NOT\_FOUND. |
| 5.2.1.3.21 | 0x931b363e, 0x8ab4, 0x49db, 0x82, 0x21, 0x2f, 0xdd, 0x9d, 0xa4, 0x36, 0x6c | RT.SetVariable – SetVariable() removes all variables with **Attributes** values of 0 at EFI\_TPL\_APPLICATION. | 1. Call SetVariable() service to insert a test variable.  2. Call SetVariable() service with **Attributes** values of 0 to delete the test variable. The returned status should be EFI\_SUCCESS.  3. Call GetVariable() service to get the test variable. The returned status should be EFI\_NOT\_FOUND. |
| 5.2.1.3.22 | 0x7eac83e5, 0x0e54, 0x4812, 0x9b, 0xb0, 0x6f, 0xf6, 0xdc, 0x7d, 0xeb, 0x8f | RT.SetVariable – SetVariable() removes all variables with **Attributes** values of 0 at EFI\_TPL\_CALLBACK. | 1. Call SetVariable() service to insert a test variable.  2. Call SetVariable() service with **Attributes** values of 0 to delete the test variable. The returned status should be EFI\_SUCCESS.  3. Call GetVariable() service to get the test variable. The returned status should be EFI\_NOT\_FOUND. |
| 5.2.1.3.23 | 0x6afdea5e, 0x1030, 0x48ab, 0x91, 0xdd, 0x7c, 0xd3, 0x53, 0x7c, 0xad, 0x3b | RT.SetVariable – checks Non-volatile variable exists after system reset at EFI\_TPL\_APPLICATION. | 1. Call SetVariable() service to insert a non-volatile test variable. The returned status must be EFI\_SUCCESS.  2. Reset the system.  3. Call GetVariable() service to get the test variable. The returned status should be EFI\_SUCCESS, and the returned data should be the same as the original data set. |
| 5.2.1.3.24 | 0x653f14cc, 0x8ecd, 0x4aaf, 0xad, 0xd6, 0x96, 0xc5, 0x07, 0x11, 0x2d, 0x67 | RT.SetVariable – checks Non-volatile variable exists after system reset at EFI\_TPL\_CALLBACK. | 1. Call SetVariable() service to insert a non-volatile test variable. The returned status must be EFI\_SUCCESS.  2. Reset the system.  3. Call GetVariable() service to get the test variable. The returned status should be EFI\_SUCCESS, and the returned data should be the same as the orginal data set. |
| 5.2.1.3.25 | 0xb93d2b03, 0x5943, 0x4c7d, 0x98, 0xec, 0xc5, 0xfe, 0x4c, 0x6e, 0x10, 0xc9 | RT.SetVariable – checks Volatile variable does not exist after system reset at EFI\_TPL\_APPLICATION. | 1. Call SetVariable() service to insert a volatile test variable. The returned status must be EFI\_SUCCESS.  2. Reset the system.  3. Call GetVariable() service to get the test variable. The returned status should be EFI\_NOT\_FOUND. |
| 5.2.1.3.26 | 0x9ec88dbe, 0xa0e4, 0x43a2, 0xaa, 0x2b, 0x60, 0xbd, 0xe6, 0xb0, 0x14, 0x1a | RT.SetVariable – Volatile variable does not exist after system reset at EFI\_TPL\_CALLBACK. | 1. Call SetVariable() service to insert a volatile test variable. The returned status must be EFI\_SUCCESS.  2. Reset the system.  3. Call GetVariable() service to get the test variable. The returned status should be EFI\_NOT\_FOUND. |
| 5.2.1.3.27 | 0x98ca8089, 0x7f55, 0x4427, 0x8c, 0x15, 0xaf, 0xa6, 0x3d, 0x78, 0x48, 0xb0 | RT.SetVariable - With DataSize is 0 | 1. Call SetVariable() service to insert a volatile test variable. The returned status must be EFI\_SUCCESS.  2. Call SetVariable() service to remove this variable with DataSize being 0. The return status should be EFI\_SUCCESS.  3. Call SetVariable() service to remove this variable with DataSize being 0. The returned status should be EFI\_NOT\_FOUND. |
| 5.2.1.3.28 | 0x008e18a5, 0xc345, 0x48ae, 0x91, 0x34, 0x61, 0xa6,  0x92, 0xe3, 0xb, 0x87 | RT.SetVariable **-** Must return EFI\_SUCCESS when creating one time-based Auth Variable. | Call SetVariable to create a time-based authenticated variable. The expected return status is EFI\_SUCCESS. |
| 5.2.1.3.29 | 0x20678b3e, 0xbcca, 0x4186, 0x84, 0xaf, 0x47, 0x16, 0xe7, 0xaf, 0xde, 0x85 | RT.SetVariable **-** The created time-based Auth Variable should pass the data validation. | Call GetVariable to retrieve the Auth Variable, and validate the Auth Variable data. |
| 5.2.1.3.30 | 0xaa6bf36f, 0xdae5, 0x43ed, 0x95, 0x4d, 0xc1, 0xc7, 0x97, 0x9d, 0x32, 0xa0 | RT.SetVariable **-**  The second Call SetVariable() with the same Data. The return status is EFI\_SECURITY\_VIOLATION. | The second Call SetVariable() with the same Data. The return status is EFI\_SECURITY\_VIOLATION. |
| 5.2.1.3.31 | 0x2bc131ec, 0x0530, 0x4994, 0xbb, 0x81, 0x15, 0x35, 0x5c, 0xef, 0xe5, 0x88 | RT.SetVariable **-** Call SetVariable()with modified/invalid Data. The expected status is EFI\_SECURITY\_VIOLATION | Call SetVariable() with modified/invalid Data. The expected status is EFI\_SECURITY\_VIOLATION  . |
| 5.2.1.3.32 | 0x0e49b21e, 0x409c, 0x4502, 0x9e, 0xc6, 0x55, 0xfe, 0x85, 0xf8, 0x54, 0x95 | RT.SetVariable - Call SetVariable() with new/valid Data. The expected status is EFI\_SUCCESS. | Call SetVariable() with new/valid Data. The expected status is EFI\_SUCCESS. |
| 5.2.1.3.33 | 0xadabac45, 0x1e0d, 0x40b0, 0x9b, 0xd1, 0x8c, 0x3a, 0xd7, 0xfb, 0x69, 0xd6 | RT.SetVariable **-** The renewed time-based Auth Variable should pass the data validation. | Call GetVariable to retrieve the renewed Auth Variable, and validate the Auth Variable data. |
| 5.2.1.3.34 | 0x6339807b, 0x0741, 0x45c4, 0x81, 0xa8, 0xe2, 0xde, 0x5a, 0x0b, 0xfb, 0x55 | RT.SetVariable **–** call SetVariable()with the old Data/timestamp. The expected status is EFI\_SECURITY\_VIOLATION. | Call SetVariable()with the old Data/timestamp. The expected status is EFI\_SECURITY\_VIOLATION |
| 5.2.1.3.35 | 0xa2d53dea, 0x8275, 0x4b9a, 0xbd, 0xa0, 0xac, 0x86, 0xfb, 0x4e, 0x0f, 0x30 | RT.SetVariable **–** call SetVariable()with the Data signed by another key, the expect status should be **EFI\_SECURITY\_VIOLATION** | Call SetVariable()with the Data signed by another key, the expect status should be EFI\_SECURITY\_VIOLATION |
| 5.2.1.3.36 | 0x28c7f0db, 0x2546, 0x4374, 0x8f, 0xf9, 0x75, 0x80, 0xc4, 0x68, 0x9b, 0x93 | RT.SetVariable **–** call SetVariable()to do the append operation, the expect status should be EFI\_SUCCESS | Call SetVariable()to do the append operation, the expect status should be EFI\_SUCCESS |
| 5.2.1.3.37 | 0x1e87dbe9, 0x234b, 0x4c82, 0x8c, 0x86, 0x2f, 0x26, 0xfa, 0xc6, 0x60, 0x2e | RT.SetVariable **–**The appended time base Auth Variable should pass the data validation | Call GetVariable()to retrieve the appended Auth Variable, and validate the Auth Variable data. |
| 5.2.1.3.38 | 0x3cc4add2, 0x0ed7, 0x4837, 0xb4, 0x63, 0xbc, 0x46, 0xd1, 0x3b, 0x2f, 0x65 | RT.SetVariable **–** call SetVariable()to do the delete operation. The expected status is EFI\_SUCCESS | Call SetVariable()to do the delete operation. The expected status is EFI\_SUCCESS |
| 5.2.1.3.39 | 0xfa50a705, 0x5d95, 0x4cad, 0xb4, 0x6c, 0xa0, 0x12, 0x9b, 0x68, 0x22, 0x8e | RT.SetVariable **–**The deleted time-based Auth Variable should not be found. | Call GetVariable()to retrieve the deleted Auth Variable. The return status should be EFI\_NOT\_FOUND. |
| 5.2.1.3.40 | 0x27e8e4de, 0x56ed, 0x4710, 0xa6, 0x3a, 0xc6, 0x35, 0xe3, 0x9d, 0x33, 0x64 | RT.SetVariable **-** must  return EFI\_SUCCESSwhen creating a time-based Auth Variable with one different key. | Call SetVariable()to create a time-based Auth Variable with one different key. The expected return status is EFI\_SUCCESS. |
| 5.2.1.3.41 | 0xba99e7f8, 0x8018, 0x46a2, 0xb2, 0xe5, 0x8b, 0xde, 0x42, 0xc1, 0xe6, 0xd5 | RT.SetVariable **–** call SetVariable()to do the append operation with the new data. The expected status is EFI\_SUCCESS | Call SetVariable()to do the append operation with the new data. The expected status is EFI\_SUCCESS |
| 5.2.1.3.42 | 0xc764906d, 0x73bb, 0x44b7, 0xae, 0x40, 0x0c, 0x51, 0xde, 0xc3, 0xc7, 0x51 | RT.SetVariable **–** call SetVariable() to set the Data with one old timestamp. The return status should be EFI\_SECURITY\_VIOLATION | Call SetVariable()to set the Data with one old timestamp. The return status should be EFI\_SECURITY\_VIOLATION |
| 5.2.1.3.43 | 0x1a28fa01, 0x135c, 0x4aeb, 0xa1, 0xb4, 0x68, 0x6a, 0x0b, 0x53, 0xb2, 0x9 | RT.SetVariable **–** call SetVariable()to do the delete operation. The expected status is EFI\_SUCCESS | Call **S**etVariable()to do the delete operation. The expected status is EFI\_SUCCESS |
| 5.2.1.3.44 | 0xe9893bcb,  0xef2b, 0x495c,  0x82, 0xf0,  0xd0, 0x63, 0x0d, 0xa7, 0x94, 0x76 | RT.SetVariable **–** must return  EFI\_SECURITY\_VIOLATION | Call SetVariable()to enroll an invalid time-based authenticated variable but several bits changed. |
| 5.2.1.3.45 | 0x2534abc0,  0x1f01, 0x48a0,  0x96, 0xde,  0xf8, 0xbb,  0xa7, 0x45,  0xc3, 0x64 | RT.SetVariable **–** must return  EFI\_SECURITY\_VIOLATION | Call SetVariable()to enroll a time-based authenticated variable with an invalid attribute. |
| 5.2.1.3.46 | 0x896f8325, 0xed28, 0x4af5, 0x96, 0xba, 0x3b, 0xe3, 0xf2, 0x97, 0x74, 0x8b | RT.SetVariable –  **SetVariable()** returns  EFI\_INVALID\_PARAMETER  When it wants to change the attribute of one existed variable  . | 1. Call **SetVariable()**  service to modify the attribute of one existed variable.  The returned status should be  **EFI\_INVALID\_PARAMETER**. |

### QueryVariableInfo()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.2.1.4.1 | 0xad9578bf, 0x7a02, 0x4ef0, 0x8f, 0xe8, 0xd9, 0x45, 0x91, 0xa1, 0xe9, 0x31 | RT.QueryVariableInfo –Query variable info with a **MaximumVariableStorageSize** value of NULL. | 1. Call QueryVariableInfo service with a **MaximumVariableStorageSize** value of NULL. The returned code must be EFI\_INVALID\_PARAMETER. |
| 5.2.1.4.2 | 0x5d13a732, 0x60ea, 0x42d5, 0xa0, 0x01, 0x43, 0x63, 0xd9, 0xb1, 0x8b, 0xf4 | RT.QueryVariableInfo –Query variable info with a **RemainingVariableStorageSize** value of NULL. | 1. Call QueryVariableInfo service with a **RemainingVariableStorageSize** value of NULL. The returned code must be EFI\_INVALID\_PARAMETER. |
| 5.2.1.4.3 | 0xd3247b73, 0x5eb9, 0x4594, 0x8a, 0xb3, 0x27, 0xd9, 0x38, 0x4f, 0x3f, 0x13 | RT.QueryVariableInfo –Query variable info with **MaximumVariableSize** value of NULL. | 1. Call QueryVariableInfo service with a **MaximumVariableSize** value of NULL. The returned code must be EFI\_INVALID\_PARAMETER. |
| 5.2.1.4.4 | 0xe7f2eb9f, 0x1624, 0x45a9, 0xa2, 0x87, 0x3e, 0xa6, 0xf2, 0xf7, 0x4c, 0x5f | RT.QueryVariableInfo –Query variable info when **Attributes** is not a combination of EFI\_VARIABLE\_RUNTIME\_ACCESS  , EFI\_VARIABLE\_BOOTSERVICE\_ACCESS  andEFI\_VARIABLE\_NON\_VOLATILE**.** | 1. Call QueryVariableInfo service with **Attributes** values of 0. The returned code must be EFI\_UNSUPPORTED. |
| 5.2.1.4.5 | 0x2f9966ba, 0x0091, 0x4085, 0xbf, 0x9d, 0x09, 0xaa, 0x80, 0x9f, 0x94, 0x2e | RT.QueryVariableInfo –Query variable info with an invalid combination of **Attributes**. | 1. Call QueryVariableInfo service with the **Attributes**:  EFI\_VARIABLE\_NON\_VOLATILE  EFI\_VARIABLE\_RUNTIME\_ACCESS  EFI\_VARIABLE\_NON\_VOLATILE|EFI\_VARIABLE\_RUNTIME\_ACCESS  The returned code must be EFI\_INVALID\_PARAMETER. |
| 5.2.1.4.8 | 0xad6e6a8f, 0x3a05, 0x4183,0xb6, 0x90, 0x40, 0xa8, 0x91, 0xd8, 0x62, 0xae | RT.QueryVariableInfo – Query variable info with a valid **Attributes** in Run time. | For each TPL less than or equal to TPL\_CALLBACK and each **Attributes** of BA, NV|BA, BA|RA and NV|BA|RA do:  1. Call QueryVariableInfo with the **Attributes** selected. Check.(Number1)  2. Call SetVariable service to insert a variable. Check.  3. Call QueryVariableInfo with the **Attributes** selected. Check. (Number2)  4. Call SetVariable service to delete the variable inserted. Check.  5. Call QueryVariableInfo service with the **Attributes** selected. Check.(Number3)  For Number1, Number2, Number3, the following items need to be checked:  1. returned codes must be EFI\_SUCCESS.  2. returned \*MaximumVariableStorageSize must be the same.  3. returned \*MaximumVariableSize must be the same, and they all are equal to MAX\_VARIABLE\_SIZE.  4. Number2 returned \*RemainingVariableStorageSize must be the value of Number1 minus the size of the variable inserted in step 2.  5. Number3 returned \*RemainingVariableStorageSize must be the value of Number1 |

### HardwareErrorRecord

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.2.1.5.1 | 0xc8126edc, 0x7197, 0x4113, 0xb7, 0xb6, 0xd5, 0x3d, 0x53, 0xe6, 0x72, 0xea | HWErrRecTest – Func Test | 1. Call GetVariable() to check theHardwareErrorRecord support of platform.  2. Call QueryVariableInfo() to detect the storage size.  3. Get a useable HWErrRec variable name and call SetVariable() to set it with data.  4. Reset system, call GetVariable() to get the data.  5. Compare the data, they should be same. |
| 5.2.1.5.2 | 0xd8bd5c0a, 0x192f, 0x4501, 0xbc, 0x58, 0x89, 0xd3, 0x18, 0x60, 0x24, 0x5e | HWErrRecTest – Conf Test HardwareErrorRecord with invalid attributes. | 1. Call GetVariable() to check theHardwareErrorRecord support of platform.  2. Call QueryVariableInfo() to detect the storage size.  3. Get a useable HWErrRec variable name and call SetVariable() to set it with invalid attributes. The returned code must be EFI\_INVALID\_PARAMETER. |
| 5.2.1.5.3 | 0xe1259932, 0xf39c, 0x465b, 0xb4, 0xe3, 0xa1, 0xb2, 0x77, 0x8b, 0xa1, 0x04 | HWErrRecTest – Conf Test HardwareErrorRecord with twice deletion. | 1. Call GetVariable() to check theHardwareErrorRecord support of platform.  2. Call QueryVariableInfo() to detect the storage size.  3. Get a useable HWErrRec variable name and call SetVariable() to set it.  4. Delete the variable twice. The first time, the returned code must be EFI\_SUCCESS; the second time, the returned code must be EFI\_INVALID\_PARAMETER. |
| 5.2.1.5.4 | 0xf5b942c9, 0x1f0c, 0x4c45, 0x85, 0x72, 0xc4, 0x53, 0x79, 0x51, 0x50, 0xdf | HWErrRecTest – Conf TestRetrive the Hardware Error Record variables, check the name of them. | 1. Call GetVariable() to check the HardwareErrorRecord support of platform. 2. Call QueryVariableInfo() to detect the storage size.3. Retrive the Hardware Error Record variables, check the name of them |

## Time Services Test

Reference Document:

*UEFI Specification,* Time Services Section

* Time Services Functions

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| GetTime | Runtime | Returns the current time and date, and the time-keeping capabilities of the platform. |
| SetTime | Runtime | Sets the current local time and date information. |
| GetWakeupTime | Runtime | Returns the current wakeup alarm clock setting. |
| SetWakeupTime | Runtime | Sets the system wakeup alarm clock time. |

### GetTime()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.2.2.1.1 | 0x105de1dc, 0x32b2, 0x4d85, 0x9b, 0x30, 0xd4, 0x41, 0x80, 0x0f, 0xdc, 0x4c | RT.GetTime – GetTime() returns EFI\_INVALID\_PARAMETER with **Time** is NULL. | 1. Call GetTime() with **Time** is NULL. The return code must be EFI\_INVALID\_PARAMETER. |
| 5.2.2.1.2 | 0x51437f55, 0x25e1, 0x43eb, 0xae, 0x76, 0x0d, 0x32, 0x1c, 0x12, 0xf6, 0x38 | RT.GetTime – GetTime() gets the system time at EFI\_TPL\_APPLICATION. | 1. Call GetTime() with valid parameters. The return code must be EFI\_SUCCESS. |
| 5.2.2.1.3 | 0x1a6e41f0, 0x361e, 0x4c46, 0xa2, 0xc4, 0x35, 0x42, 0xb3, 0x6f, 0xa5, 0xb6 | RT.GetTime – GetTime() gets the system time at EFI\_TPL\_CALLBACK. | 1. Call GetTime() with valid parameters. The return code must be EFI\_SUCCESS. |
| 5.2.2.1.4 | 0x3568b497, 0x6524, 0x4415, 0xac, 0xaa, 0xa8, 0xee, 0x24, 0x83, 0x9b, 0xdd | RT.GetTime – GetTime() gets the system time at EFI\_TPL\_APPLICATION. | 1. Call GetTime() with valid parameters. The return time should be valid. |
| 5.2.2.1.5 | 0xa2c13016, 0x01d4, 0x4ea7, 0xb0, 0x8e, 0xb7, 0x74, 0x22, 0x4d, 0x7e, 0xa5 | RT.GetTime – GetTime() gets the system time at EFI\_TPL\_CALLBACK. | 1. Call GetTime() with valid parameters. The return time should be valid. |
| 5.2.2.1.6 | 0x2cd14974, 0x4937, 0x4817, 0x91, 0xb0, 0x82, 0x2f, 0x40, 0xca, 0x22, 0xbc | RT.GetTime – GetTime() gets the system time with **Capabilities** is NULL at EFI\_TPL\_APPLICATION. | 1. Call GetTime() with a **Capabilities** value of NULL. The return code should be EFI\_SUCCESS. |
| 5.2.2.1.7 | 0x9bbabc14, 0xced2, 0x48fc, 0xbb, 0x9e, 0x79, 0x37, 0x49, 0xe8, 0x1f, 0xe2 | RT.GetTime – GetTime() gets the system time with **Capabilities** is NULL at EFI\_TPL\_CALLBACK. | 1. Call GetTime() with a **Capabilities** value of NULL. The return code should be EFI\_SUCCESS. |
| 5.2.2.1.8 | 0x938366e9, 0x3311, 0x4007, 0x87, 0xc3, 0xa2, 0x18, 0x7f, 0x05, 0x14, 0xe3 | RT.GetTime – GetTime() gets the system time with **Capabilities** is NULL at EFI\_TPL\_APPLICATION. | 1. Call GetTime() with a **Capabilities** value of NULL. The return time should be valid. |
| 5.2.2.1.9 | 0x565f4b15, 0xb132, 0x4c74, 0x97, 0xc2, 0xf3, 0xa6, 0xf5, 0xbf, 0xd2, 0x21 | RT.GetTime – GetTime() gets the system time with **Capabilities** is NULL at EFI\_TPL\_CALLBACK. | 1. Call GetTime() with a **Capabilities** value of NULL. The return time should be valid. |

### SetTime()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.2.2.2.1 | 0x6f96cde3, 0x6067, 0x4213, 0x81, 0xf8, 0x45, 0x90, 0x1d, 0x92, 0x1a, 0x12 | RT.SetTime - SetTime() returns EFI\_INVALID\_PARAMETER with **Year** is less than the low range. | 1. Call SetTime() with **Time.Year** is 1899. The return code must be EFI\_INVALID\_PARAMETER. |
| 5.2.2.2.2 | 0x8ce9f594, 0x2d49, 0x4436, 0xb1, 0xd1, 0xe4, 0xd4, 0xbf, 0x55, 0x41, 0xdc | RT.SetTime - SetTime() returns EFI\_INVALID\_PARAMETER with **Year** is greater than the upper range. | 1. Call SetTime() with **Time.Year** is 10000. The return code must be EFI\_INVALID\_PARAMETER. |
| 5.2.2.2.3 | 0x972fadc8, 0x5cc4, 0x4cbe, 0xbe, 0xd6, 0x76, 0xca, 0xef, 0x2d, 0x1b, 0x1a | RT.SetTime - SetTime() returns EFI\_INVALID\_PARAMETER with **Year** is invalid. | 1. Call SetTime() with **Time.Year** is -1. The return code must be EFI\_INVALID\_PARAMETER. |
| 5.2.2.2.4 | 0xcaac8a85, 0x26c2, 0x43e7, 0x83, 0x40, 0x5a, 0x78, 0x85, 0x43, 0xef, 0x81 | RT.SetTime - SetTime() returns EFI\_INVALID\_PARAMETER with **Month** is less than the low range. | 1. Call SetTime() with **Time.Month** is 0. The return code must be EFI\_INVALID\_PARAMETER. |
| 5.2.2.2.5 | 0x12470ee0, 0x19e1, 0x49ff, 0xbc, 0x1e, 0x8e, 0xb3, 0x6f, 0xab, 0xf0, 0xfc | RT.SetTime - SetTime() returns EFI\_INVALID\_PARAMETER with **Month** is greater than the upper range. | 1. Call SetTime() with **Time.Month** is 13. The return code must be EFI\_INVALID\_PARAMETER. |
| 5.2.2.2.6 | 0xae7293c9, 0x0cbd, 0x4317, 0xb6, 0xeb, 0x33, 0xe1, 0x83, 0x46, 0x8d, 0x9e | RT.SetTime - SetTime() returns EFI\_INVALID\_PARAMETER with **Month** is invalid. | 1. Call SetTime() with **Time.Month** is -1. The return code must be EFI\_INVALID\_PARAMETER. |
| 5.2.2.2.7 | 0xb8048c3c, 0xbf1f, 0x477d, 0xb7, 0x17, 0x55, 0x41, 0xfc, 0xa7, 0xb5, 0x61 | RT.SetTime - SetTime() returns EFI\_INVALID\_PARAMETER with **Day** is less than the low range. | 1. Call SetTime() with **Time.Day** is 0. The return code must be EFI\_INVALID\_PARAMETER. |
| 5.2.2.2.8 | 0x0d2c6265, 0xad3a, 0x4554, 0xb0, 0x16, 0x6c, 0xb7, 0xff, 0x59, 0x1f, 0x78 | RT.SetTime - SetTime() returns EFI\_INVALID\_PARAMETER with **Day** is greater than the upper range. | 1. Call SetTime() with **Time.Day** is 32. The return code must be EFI\_INVALID\_PARAMETER. |
| 5.2.2.2.9 | 0x0467b0c4, 0xdf8c, 0x4bfc, 0xa8, 0x4b, 0xef, 0xa6, 0x90, 0x5b, 0xde, 0xd9 | RT.SetTime - SetTime() returns EFI\_INVALID\_PARAMETER with **Day** is invalid. | 1. Call SetTime() with **Time.Day** is -1. The return code must be EFI\_INVALID\_PARAMETER. |
| 5.2.2.2.10 | 0x1e433b44, 0xa599, 0x4dcd, 0x9c, 0x38, 0xe7, 0xc0, 0x97, 0xf2, 0x56, 0x4b | RT.SetTime - SetTime() returns EFI\_INVALID\_PARAMETER with **Day** is greater than the upper range. | 1. Call SetTime() with **Time.Month** is 4 and **Time.Day** is 31. The return code must be EFI\_INVALID\_PARAMETER. |
| 5.2.2.2.11 | 0xc9bfb088, 0x07ba, 0x413c, 0xa4, 0x72, 0xbd, 0x17, 0x92, 0xdd, 0xc6, 0xec | RT.SetTime - SetTime() returns EFI\_INVALID\_PARAMETER with **Hour** is greater than the upper range. | 1. Call SetTime() with **Time.Hour** is 24. The return code must be EFI\_INVALID\_PARAMETER. |
| 5.2.2.2.12 | 0xd7b3ca07, 0xa484, 0x4604, 0x83, 0x37, 0x6f, 0x13, 0x4f, 0x88, 0xb3, 0x5a | RT.SetTime - SetTime() returns EFI\_INVALID\_PARAMETER with **Hour** is invalid. | 1. Call SetTime() with **Time.Hour** is -1. The return code must be EFI\_INVALID\_PARAMETER. |
| 5.2.2.2.13 | 0xc645baaa, 0x3eb6, 0x4577, 0x97, 0x5d, 0x21, 0x05, 0x04, 0x83, 0x64, 0x2b | RT.SetTime - SetTime() returns EFI\_INVALID\_PARAMETER with **Minute** is greater than the upper range. | 1. Call SetTime() with **Time.Minute** is 60. The return code must be EFI\_INVALID\_PARAMETER. |
| 5.2.2.2.14 | 0xa42f7c8e, 0xfa7a, 0x4026, 0xb9, 0x6b, 0x66, 0xe3, 0xf2, 0xe9, 0x93, 0x55 | RT.SetTime - SetTime() returns EFI\_INVALID\_PARAMETER with **Minute** is invalid. | 1. Call SetTime() with **Time.Minute** is -1. The return code must be EFI\_INVALID\_PARAMETER. |
| 5.2.2.2.15 | 0xd37d5f03, 0x6dbb, 0x4724, 0x9e, 0xc1, 0xed, 0x13, 0x6b, 0x17, 0x22, 0xe9 | RT.SetTime - SetTime() returns EFI\_INVALID\_PARAMETER with **Second** is greater than the upper range. | 1. Call SetTime() with **Time.Second** is 60. The return code must be EFI\_INVALID\_PARAMETER. |
| 5.2.2.2.16 | 0xcd47c7aa, 0x6522, 0x45ed, 0xa7, 0xb4, 0x29, 0x6d, 0x57, 0x43, 0xc7, 0x78 | RT.SetTime - SetTime() returns EFI\_INVALID\_PARAMETER with **Second** is invalid. | 1. Call SetTime() with **Time.Second** is -1. The return code must be EFI\_INVALID\_PARAMETER. |
| 5.2.2.2.17 | 0x14bccf9f, 0xda75, 0x46db, 0xb1, 0xfc, 0x7e, 0x67, 0x3b, 0x37, 0x25, 0x6e | RT.SetTime - SetTime() returns EFI\_INVALID\_PARAMETER with **Nanosecond** is greater than the upper range. | 1. Call SetTime() with **Time.Nanosecond** is 1000000000. The return code must be EFI\_INVALID\_PARAMETER. |
| 5.2.2.2.18 | 0x966cf8d6, 0xf952, 0x4770, 0xa1, 0x9e, 0xf8, 0x78, 0xbc, 0x60, 0xbc, 0xeb | RT.SetTime - SetTime() returns EFI\_INVALID\_PARAMETER with **Nanosecond** is invalid. | 1. Call SetTime() with **Time.Nanosecond** is -1. The return code must be EFI\_INVALID\_PARAMETER. |
| 5.2.2.2.19 | 0x59a9febb, 0xf6d1, 0x4b13, 0xae, 0xcd, 0xf3, 0x65, 0xc2, 0x11, 0xa4, 0xed | RT.SetTime - SetTime() returns EFI\_INVALID\_PARAMETER with **TimeZone** is less than the low range. | 1. Call SetTime() with **Time.TimeZone** is -1441. The return code must be EFI\_INVALID\_PARAMETER. |
| 5.2.2.2.20 | 0x5786f2c1, 0x48a7, 0x4856, 0x89, 0xe7, 0xba, 0xce, 0xc0, 0x85, 0xf3, 0xf9 | RT.SetTime - SetTime() returns EFI\_INVALID\_PARAMETER with **TimeZone** is greater than the upper range. | 1. Call SetTime() with **Time.TimeZone** is 1441. The return code must be EFI\_INVALID\_PARAMETER. |
| 5.2.2.2.21 | 0xd3a1cbdd, 0x1df5, 0x4d24, 0x97, 0x53, 0xc3, 0xae, 0xa2, 0x7a, 0xab, 0x46 | RT.SetTime – SetTime() returns EFI\_INVALID\_PARAMETER with invalid leap day. | 1. Call SetTime() with **Time** is 2001/2/29. The return code must be EFI\_INVALID\_PARAMETER. |
| 5.2.2.2.22 | 0x29151ae4, 0x7a5e, 0x42d9, 0x84, 0xf8, 0xe9, 0xc5, 0x67, 0x87, 0xb7, 0xe8 | RT.SetTime – SetTime() returns EFI\_SUCCESS to update the **Year** at EFI\_TPL\_APPLICATION. | 1. Call SetTime() to update the **Time.Year**. The return code should be EFI\_SUCCESS. |
| 5.2.2.2.23 | 0x75e988ee, 0xec78, 0x4190, 0x9a, 0x09, 0xb1, 0x31, 0x5c, 0x20, 0x25, 0xa5 | RT.SetTime – SetTime() returns EFI\_SUCCESS to update the **Year** at EFI\_TPL\_CALLBACK. | 1. Call SetTime() to update the **Time.Year**. The return code should be EFI\_SUCCESS. |
| 5.2.2.2.24 | 0x3b96a20c, 0x2b1f, 0x44ea, 0xba, 0xa9, 0xf9, 0x6f, 0xee, 0x13, 0x1d, 0x05 | RT.SetTime – SetTime() returns EFI\_SUCCESS to update the **Year** at EFI\_TPL\_APPLICATION. | 1. Call SetTime() to update the **Time.Year**.  2. Call GetTime() to verify the updated **Year**. The return **Time** should be set before. |
| 5.2.2.2.25 | 0xe664e1d7, 0xb733, 0x410d, 0xbc, 0x53, 0xd4, 0xcf, 0xf2, 0x46, 0x43, 0x55 | RT.SetTime – SetTime() returns EFI\_SUCCESS to update the **Year** at EFI\_TPL\_CALLBACK. | 1. Call SetTime() to update the **Time.Year**.  2. Call GetTime() to verify the updated **Year**. The return **Time** should be set before. |
| 5.2.2.2.26 | 0x4e123824, 0x8636, 0x4426, 0x81, 0xe6, 0x16, 0x75, 0x62, 0x8c, 0xde, 0x69 | RT.SetTime – SetTime() returns EFI\_SUCCESS to update the **Month** at EFI\_TPL\_APPLICATION. | 1. Call SetTime() to update the **Time.Month**. The return code should be EFI\_SUCCESS. |
| 5.2.2.2.27 | 0x8f0bfe23, 0xb6ec, 0x4ea2, 0x8e, 0x03, 0x0a, 0x7a, 0x5e, 0x36, 0x45, 0xb3 | RT.SetTime – SetTime() returns EFI\_SUCCESS to update the **Month** at EFI\_TPL\_CALLBACK. | 1. Call SetTime() to update the **Time.Month**. The return code should be EFI\_SUCCESS. |
| 5.2.2.2.28 | 0x2d5cdbe5, 0x1055, 0x4ef6, 0x8e, 0x90, 0x0c, 0x99, 0x3f, 0x93, 0xf6, 0x98 | RT.SetTime – SetTime() returns EFI\_SUCCESS to update the **Month** at EFI\_TPL\_APPLICATION. | 1. Call SetTime() to update the **Time.Month**.  2. Call GetTime() to verify the updated **Month**. The return **Time** should be set before. |
| 5.2.2.2.29 | 0xda4b19e7, 0xf605, 0x4fb9, 0xa1, 0x81, 0xcc, 0xd3, 0x35, 0x29, 0x0b, 0xfe | RT.SetTime – SetTime() returns EFI\_SUCCESS to update the **Month** at EFI\_TPL\_CALLBACK. | 1. Call SetTime() to update the **Time.Month**.  2. Call GetTime() to verify the updated **Month**. The return **Time** should be set before. |
| 5.2.2.2.30 | 0x7af90ce7, 0x1fed, 0x4101, 0x82, 0xdc, 0xcc, 0x63, 0x4c, 0xdf, 0x20, 0x4e | RT.SetTime – SetTime() returns EFI\_SUCCESS to update the daylight at EFI\_TPL\_APPLICATION. | 1. Call SetTime() to update the **Time.Daylight**. The return code should be EFI\_SUCCESS. |
| 5.2.2.2.31 | 0xfa81d174, 0x5743, 0x485f, 0xb2, 0x48, 0xaa, 0xea, 0xdd, 0x7c, 0x1e, 0x51 | RT.SetTime – SetTime() returns EFI\_SUCCESS to update the daylight at EFI\_TPL\_CALLBACK. | 1. Call SetTime() to update the **Time.Daylight**. The return code should be EFI\_SUCCESS. |
| 5.2.2.2.32 | 0xb39bc904, 0x55e7, 0x4b9b, 0xb4, 0xd8, 0x27, 0x4a, 0xdd, 0x71, 0xd6, 0x25 | RT.SetTime – SetTime() returns EFI\_SUCCESS to update the daylight at EFI\_TPL\_APPLICATION. | 1. Call SetTime() to update the **Time.Daylight**.  2. Call GetTime() to verify the updated **Daylight**. The return **Time** should be set before. |
| 5.2.2.2.33 | 0x54daf29b, 0x48e6, 0x4fa4, 0xad, 0x00, 0xb8, 0xd6, 0x48, 0xaf, 0x7d, 0x88 | RT.SetTime – SetTime() returns EFI\_SUCCESS to update the daylight at EFI\_TPL\_CALLBACK. | 1. Call SetTime() to update the **Time.Daylight**.  2. Call GetTime() to verify the updated **Daylight**. The return **Time** should be set before. |
| 5.2.2.2.34 | 0xcdbbda04, 0x4f7c, 0x4ba5, 0x8b, 0xcf, 0xc0, 0x50, 0xe5, 0xa9, 0x76, 0xc7 | RT.SetTime – SetTime() returns EFI\_SUCCESS to update the **TimeZone** at EFI\_TPL\_APPLICATION. | 1. Call SetTime() to update the **Time.TimeZone**. The return code should be EFI\_SUCCESS. |
| 5.2.2.2.35 | 0xf749b4f1, 0x537d, 0x4ddf, 0x85, 0x45, 0xc0, 0xa4, 0x19, 0x93, 0xce, 0xe4 | RT.SetTime – SetTime() returns EFI\_SUCCESS to update the **TimeZone** at EFI\_TPL\_CALLBACK. | 1. Call SetTime() to update the **Time.TimeZone**. The return code should be EFI\_SUCCESS. |
| 5.2.2.2.36 | 0xea99dec5, 0xb879, 0x4c8d, 0xbf, 0xd1, 0xf6, 0x3f, 0xe7, 0x58, 0x99, 0xbf | RT.SetTime – SetTime() returns EFI\_SUCCESS to update the **TimeZone** at EFI\_TPL\_APPLICATION. | 1. Call SetTime() to update the **Time.TimeZone**.  2. Call GetTime() to verify the updated **TimeZone**. The return **Time** should be set before. |
| 5.2.2.2.37 | 0xd9c645b9, 0x52de, 0x415c, 0xab, 0xdc, 0x72, 0x26, 0xce, 0x6a, 0x30, 0xb1 | RT.SetTime – SetTime() returns EFI\_SUCCESS to update the **TimeZone** at EFI\_TPL\_CALLBACK. | 1. Call SetTime() to update the **Time.TimeZone**.  2. Call GetTime() to verify the updated **TimeZone**. The return **Time** should be set before. |

### 

### GetWakeupTime()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.2.2.3.1 | 0xbb9fd931, 0xd3c0, 0x43cd, 0xb0, 0xa7, 0xfe, 0x17, 0xdc, 0xd7, 0x4d, 0x53 | RT.GetWakeupTime – GetWakeupTime() returns EFI\_INVALID\_PARAMETER with **Enabled** is NULL. | 1. Call GetWakeupTime() with **Enabled** is NULL. The return code must be EFI\_UNSUPPORTED or EFI\_INVALID\_PARAMETER. |
| 5.2.2.3.2 | 0x200b6e00, 0x9e1b, 0x4891, 0x83, 0x01, 0xef, 0x46, 0x9f, 0x31, 0x17, 0x08 | RT.GetWakeupTime – GetWakeupTime() returns EFI\_INVALID\_PARAMETER with **Pending** is NULL. | 1. Call GetWakeupTime() with **Pending** is NULL. The return code must be EFI\_UNSUPPORTED or EFI\_INVALID\_PARAMETER. |
| 5.2.2.3.3 | 0x209435c5, 0xfa4f, 0x405d, 0x80, 0xa6, 0x9e, 0xdc, 0x9d, 0x38, 0x8c, 0xc6 | RT.GetWakeupTime – GetWakeupTime() returns EFI\_INVALID\_PARAMETER with **Time** is NULL. | 1. Call GetWakeupTime() with **Time** is NULL. The return code must be EFI\_UNSUPPORTED or EFI\_INVALID\_PARAMETER. |
| 5.2.2.3.4 | 0xe553c375, 0xd529, 0x4610, 0xad, 0xb5, 0x3a, 0x56, 0xc3, 0xec, 0xcb, 0xe9 | RT.GetWakeupTime – GetWakeupTime() returns EFI\_SUCCESS at EFI\_TPL\_APPLICATION. | 1. Call GetWakeupTime() with valid parameters. The return code must be EFI\_UNSUPPORTED or EFI\_SUCCESS. |
| 5.2.2.3.5 | 0x36414d2a, 0xf932, 0x43ca, 0xab, 0x08, 0x41, 0x8e, 0x59, 0xd9, 0xa4, 0xa2 | RT.GetWakeupTime – GetWakeupTime() returns EFI\_SUCCESS at EFI\_TPL\_CALLBACK. | 1. Call GetWakeupTime() with valid parameters. The return code must be EFI\_UNSUPPORTED or EFI\_SUCCESS. |
| 5.2.2.3.6 | 0x6092de6c, 0x062f, 0x4adb, 0xab, 0x4b, 0xb4, 0xda, 0x69, 0xd2, 0x8e, 0xd8 | RT.GetWakeupTime – GetWakeupTime() gets the wakeup status at EFI\_TPL\_APPLICATION. | 1. Call GetWakeupTime() with valid parameters. If the **Enabled** is TRUE, the return time should be valid. |
| 5.2.2.3.7 | 0x8061bae9, 0x341c, 0x48ab, 0xad, 0x37, 0x15, 0x5c, 0x6b, 0x0f, 0x13, 0x34 | RT.GetWakeupTime – GetWakeupTime() gets the wakeup status at EFI\_TPL\_CALLBACK. | 1. Call GetWakeupTime() with valid parameters. If the **Enabled** is TRUE, the return time should be valid. |

### SetWakeupTime()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.2.2.4.1 | 0x41d27daf, 0xe088, 0x441c, 0xb2, 0x05, 0x6d, 0xd7, 0xa4, 0xac, 0x08, 0xb1 | RT.SetWakeupTime – SetWakeupTime() returns EFI\_INVALID\_PARAMETER with **Year** is less than the low range. | 1. Call SetWakeupTime() with **Time.Year** is 1997. The return code must be EFI\_UNSUPPORTED or EFI\_INVALID\_PARAMETER. |
| 5.2.2.4.2 | 0xe2dbc697, 0xc56a, 0x4c58, 0xa2, 0x74, 0x58, 0x99, 0x94, 0x1c, 0x7e, 0x02 | RT.SetWakeupTime – SetWakeupTime() returns EFI\_INVALID\_PARAMETER with **Year** is greater than the upper range. | 1. Call SetWakeupTime() with **Time.Year** is 2100. The return code must be EFI\_UNSUPPORTED or EFI\_INVALID\_PARAMETER. |
| 5.2.2.4.3 | 0x2ef795b9, 0xdfac, 0x4334, 0xa2, 0x43, 0x55, 0xbe, 0x0d, 0x0c, 0x3b, 0x44 | RT.SetWakeupTime – SetWakeupTime() returns EFI\_INVALID\_PARAMETER with **Year** is invalid. | 1. Call SetWakeupTime() with **Time.Year** is -1. The return code must be EFI\_UNSUPPORTED or EFI\_INVALID\_PARAMETER. |
| 5.2.2.4.4 | 0x8f7fe2f6, 0xd96d, 0x4765, 0x96, 0x42, 0x05, 0xae, 0x30, 0x66, 0xd8, 0xb9 | RT.SetWakeupTime – SetWakeupTime() returns EFI\_INVALID\_PARAMETER with **Month** is less than the low range. | 1. Call SetWakeupTime() with **Time.Month** is 0. The return code must be EFI\_UNSUPPORTED or EFI\_INVALID\_PARAMETER. |
| 5.2.2.4.5 | 0xc398668f, 0x03c2, 0x4cac, 0x81, 0x18, 0x7c, 0xbe, 0xab, 0xd1, 0xb9, 0x67 | RT.SetWakeupTime – SetWakeupTime() returns EFI\_INVALID\_PARAMETER with **Month** is greater than the upper range. | 1. Call SetWakeupTime() with **Time.Month** is 13. The return code must be EFI\_UNSUPPORTED or EFI\_INVALID\_PARAMETER. |
| 5.2.2.4.6 | 0x57a4eedd, 0xafa6, 0x4233, 0xb2, 0xeb, 0x79, 0xe4, 0x5e, 0x3d, 0xc0, 0x2d | RT.SetWakeupTime – SetWakeupTime() returns EFI\_INVALID\_PARAMETER with **Month** is invalid. | 1. Call SetWakeupTime() with **Time.Month** is -1. The return code must be EFI\_UNSUPPORTED or EFI\_INVALID\_PARAMETER. |
| 5.2.2.4.7 | 0x61dd2e73, 0x0c29, 0x436a, 0x80, 0x73, 0x3c, 0xe4, 0xde, 0xc7, 0x0d, 0xf2 | RT.SetWakeupTime – SetWakeupTime() returns EFI\_INVALID\_PARAMETER with **Day** is less than the low range. | 1. Call SetWakeupTime() with **Time.Day** is 0. The return code must be EFI\_UNSUPPORTED or EFI\_INVALID\_PARAMETER. |
| 5.2.2.4.8 | 0x7c532de7, 0x3d59, 0x4a43, 0x9c, 0xf1, 0x8c, 0x35, 0x51, 0x70, 0xbc, 0x86 | RT.SetWakeupTime – SetWakeupTime() returns EFI\_INVALID\_PARAMETER with **Day** is greater than the upper range. | 1. Call SetWakeupTime() with **Time.Day** is 32. The return code must be EFI\_UNSUPPORTED or EFI\_INVALID\_PARAMETER. |
| 5.2.2.4.9 | 0xb07ea402, 0x8403, 0x4c42, 0xa4, 0x11, 0x23, 0x2c, 0x37, 0xf9, 0xc5, 0x27 | RT.SetWakeupTime – SetWakeupTime() returns EFI\_INVALID\_PARAMETER with **Day** is invalid. | 1. Call SetWakeupTime() with **Time.Day** is -1. The return code must be EFI\_UNSUPPORTED or EFI\_INVALID\_PARAMETER. |
| 5.2.2.4.10 | 0xc86e5f11, 0x2e97, 0x4cee, 0x9c, 0xc8, 0xd3, 0xf5, 0x7f, 0xa6, 0x46, 0x75 | RT.SetWakeupTime – SetWakeupTime() returns EFI\_INVALID\_PARAMETER with **Day** is greater than the upper range. | 1. Call SetWakeupTime() with **Time.Month** is 4 and **Time.Day** is 31. The return code must be EFI\_UNSUPPORTED or EFI\_INVALID\_PARAMETER. |
| 5.2.2.4.11 | 0x0ef3f79c, 0x9399, 0x47f8, 0xab, 0x3b, 0xa6, 0x6c, 0x2f, 0x78, 0x1f, 0x9e | RT.SetWakeupTime – SetWakeupTime() returns EFI\_INVALID\_PARAMETER with **Hour** is greater than the upper range. | 1. Call SetWakeupTime() with **Time.Hour** is 24. The return code must be EFI\_UNSUPPORTED or EFI\_INVALID\_PARAMETER. |
| 5.2.2.4.12 | 0x9f61f3ac, 0x059b, 0x4658, 0x98, 0x2d, 0x61, 0x6e, 0xab, 0x25, 0xcb, 0x6d | RT.SetWakeupTime – SetWakeupTime() returns EFI\_INVALID\_PARAMETER with **Hour** is invalid. | 1. Call SetWakeupTime() with **Time.Hour** is -1. The return code must be EFI\_UNSUPPORTED or EFI\_INVALID\_PARAMETER. |
| 5.2.2.4.13 | 0xa05b10e8, 0x098e, 0x4c02, 0xad, 0x30, 0xef, 0xac, 0x58, 0xf4, 0x07, 0x56 | RT.SetWakeupTime – SetWakeupTime() returns EFI\_INVALID\_PARAMETER with **Minute** is greater than the upper range. | 1. Call SetWakeupTime() with **Time.Minute** is 60. The return code must be EFI\_UNSUPPORTED or EFI\_INVALID\_PARAMETER. |
| 5.2.2.4.14 | 0xbca1c0cf, 0xe121, 0x42fc, 0xba, 0x49, 0x2b, 0xd0, 0xad, 0x74, 0x3d, 0x60 | RT.SetWakeupTime – SetWakeupTime() returns EFI\_INVALID\_PARAMETER with **Minute** is invalid. | 1. Call SetWakeupTime() with **Time.Minute** is -1. The return code must be EFI\_UNSUPPORTED or EFI\_INVALID\_PARAMETER. |
| 5.2.2.4.15 | 0x89c7e1f1, 0x98cb, 0x4f3c, 0x96, 0xc7, 0x03, 0x59, 0x22, 0xd0, 0xce, 0x34 | RT.SetWakeupTime – SetWakeupTime() returns EFI\_INVALID\_PARAMETER with **Second** is greater than the upper range. | 1. Call SetWakeupTime() with **Time.Second** is 60. The return code must be EFI\_UNSUPPORTED or EFI\_INVALID\_PARAMETER. |
| 5.2.2.4.16 | 0x59b0d53d, 0xffac, 0x4c1a, 0xb9, 0xb0, 0x2c, 0xe6, 0xfc, 0x93, 0x8f, 0x0e | RT.SetWakeupTime – SetWakeupTime() returns EFI\_INVALID\_PARAMETER with **Second** is invalid. | 1. Call SetWakeupTime() with **Time.Second** is -1. The return code must be EFI\_UNSUPPORTED or EFI\_INVALID\_PARAMETER. |
| 5.2.2.4.17 | 0x98737393, 0x45af, 0x4945, 0xa7, 0xd2, 0xe2, 0x92, 0xfd, 0x4e, 0x8d, 0x20 | RT.SetWakeupTime – SetWakeupTime() returns EFI\_INVALID\_PARAMETER with **Nanosecond** is greater than the upper range. | 1. Call SetWakeupTime() with **Time.Nanosecond** is 1000000000. The return code must be EFI\_UNSUPPORTED or EFI\_INVALID\_PARAMETER. |
| 5.2.2.4.18 | 0xc9eff904, 0x5d44, 0x451c, 0x94, 0xd2, 0x66, 0x73, 0xe1, 0x8e, 0x65, 0x05 | RT.SetWakeupTime – SetWakeupTime() returns EFI\_INVALID\_PARAMETER with **Nanosecond** is invalid. | 1. Call SetWakeupTime() with **Time.Nanosecond** is -1. The return code must be EFI\_UNSUPPORTED or EFI\_INVALID\_PARAMETER. |
| 5.2.2.4.19 | 0x4cf4b039, 0xf2aa, 0x4f8a, 0x9c, 0xec, 0x0a, 0x80, 0x2c, 0xea, 0xd7, 0x5f | RT.SetWakeupTime – SetWakeupTime() returns EFI\_INVALID\_PARAMETER with **TimeZone** is less than the low range. | 1. Call SetWakeupTime() with **Time.TimeZone** is -1441. The return code must be EFI\_UNSUPPORTED or EFI\_INVALID\_PARAMETER. |
| 5.2.2.4.20 | 0xabd093eb, 0x7d84, 0x4ebc, 0xb3, 0x24, 0xc2, 0x85, 0x79, 0x5b, 0xde, 0x34 | RT.SetWakeupTime – SetWakeupTime() returns EFI\_INVALID\_PARAMETER with **TimeZone** is greater than the upper range. | 1. Call SetWakeupTime() with **Time.TimeZone** is 1441. The return code must be EFI\_UNSUPPORTED or EFI\_INVALID\_PARAMETER. |
| 5.2.2.4.21 | 0x0fce1f4c, 0x41f6, 0x4de4, 0x80, 0xa7, 0x77, 0x14, 0xa0, 0x35, 0x6d, 0x9b | RT.SetWakeupTime – SetWakeupTime() returns EFI\_INVALID\_PARAMETER with invalid leap day. | 1. Call SetWakeupTime() with **Time** is 2001/2/29. The return code must be EFI\_UNSUPPORTED or EFI\_INVALID\_PARAMETER. |
| 5.2.2.4.22 | 0x4b660fec, 0xc2d0, 0x423f, 0xa3, 0x87, 0x07, 0x80, 0x41, 0xa1, 0x83, 0xb7 | RT.SetWakeupTime – SetWakeupTime() returns EFI\_SUCCESS with valid parameters at EFI\_TPL\_APPLICATION. | 1. Call SetWakeupTime() with valid parameters. The return code should be EFI\_SUCCESS. |
| 5.2.2.4.23 | 0x218d16a6, 0xf52a, 0x4e42, 0x80, 0x52, 0x1a, 0x4d, 0x5d, 0x4a, 0x19, 0x60 | RT.SetWakeupTime – SetWakeupTime() returns EFI\_SUCCESS with valid parameters at EFI\_TPL\_CALLBACK. | 1. Call SetWakeupTime() with valid parameters. The return code should be EFI\_SUCCESS. |
| 5.2.2.4.24 | 0x0da0ec8a, 0xb748, 0x4c42, 0xa8, 0xc6, 0x71, 0x03, 0x75, 0x32, 0x90, 0x71 | RT.SetWakeupTime – SetWakeupTime() enables the wakeup time at EFI\_TPL\_APPLICATION. | 1. Call SetWakeupTime() with valid parameters.  2. Call GetWakeupTime() to get the wakeup time. The return **Enabled** should be TRUE. |
| 5.2.2.4.25 | 0x34aaf995, 0xd29b, 0x4892, 0xa4, 0x18, 0x99, 0x2c, 0xb0, 0xee, 0x29, 0xea | RT.SetWakeupTime – SetWakeupTime() enables the wakeup time at EFI\_TPL\_CALLBACK. | 1. Call SetWakeupTime() with valid parameters.  2. Call GetWakeupTime() to get the wakeup time. The return **Enabled** should be TRUE. |
| 5.2.2.4.26 | 0x49f3c56e, 0x013b, 0x4fa8, 0x8a, 0xb2, 0x17, 0x70, 0xd5, 0x37, 0x3d, 0x74 | RT.SetWakeupTime – SetWakeupTime() enables the wakeup time at EFI\_TPL\_APPLICATION. | 1. Call SetWakeupTime() with valid parameters.  2. Call GetWakeupTime() to get the wakeup time. The return **Pending** should be FALSE. |
| 5.2.2.4.27 | 0xb39225e6,  0x3d06, 0x401c, 0xad, 0x26, 0x3e, 0xa9, 0x23, 0x71, 0xf3, 0xdc | RT.SetWakeupTime – SetWakeupTime() enables the wakeup time at EFI\_TPL\_CALLBACK. | 1. Call SetWakeupTime() with valid parameters.  2. Call GetWakeupTime() to get the wakeup time. The return **Pending** should be FALSE. |
| 5.2.2.4.28 | 0x6fd3d6d4, 0x2694, 0x4677, 0x87, 0x76, 0x3d, 0xd6, 0x2e, 0x3a, 0x8c, 0xa0 | RT.SetWakeupTime – SetWakeupTime() enables the wakeup time at EFI\_TPL\_APPLICATION. | 1. Call SetWakeupTime() with valid parameters.  2. Call GetWakeupTime() to get the wakeup time. The return **Time** should be set before. |
| 5.2.2.4.29 | 0xdf714d88, 0x9ee9, 0x4027, 0xa3, 0x70, 0xe5, 0xa2, 0x83, 0x56, 0x5c, 0xed | RT.SetWakeupTime – SetWakeupTime() enables the wakeup time at EFI\_TPL\_CALLBACK. | 1. Call SetWakeupTime() with valid parameters.  2. Call GetWakeupTime() to get the wakeup time. The return **Time** should be set before. |
| 5.2.2.4.30 | 0xd3835a5c, 0xb4be, 0x4f6c, 0xab, 0xf0, 0x29, 0x52, 0x52, 0x37, 0x14, 0x06 | RT.SetWakeupTime – SetWakeupTime() disables the wakeup time with **Enable** is FALSE at EFI\_TPL\_APPLICATION. | 1. Call SetWakeupTime() with **Enable** is FALSE. The return code must be EFI\_SUCCESS. |
| 5.2.2.4.31 | 0xeb8730ec, 0x578d, 0x41b1, 0xa2, 0xbe, 0x4a, 0x9f, 0xf6, 0x03, 0xdb, 0x22 | RT.SetWakeupTime – SetWakeupTime() disables the wakeup time with **Enable** is FALSE at EFI\_TPL\_CALLBACK. | 1. Call SetWakeupTime() with **Enable** is FALSE. The return code must be EFI\_SUCCESS. |
| 5.2.2.4.32 | 0xffaa1029, 0x16ae, 0x4d5c, 0xba, 0x74, 0x86, 0x80, 0xf4, 0xba, 0x9c, 0xd0 | RT.SetWakeupTime – SetWakeupTime() disables the wakeup time with **Enable** is FALSE at EFI\_TPL\_APPLICATION. | 1. Call SetWakeupTime() with **Enable** is FALSE.  2. Call GetWakeupTime() to get the wakeup time. The return **Enabled** must be FALSE. |
| 5.2.2.4.33 | 0x8a70609a, 0xab54, 0x475e, 0x8d, 0xf2, 0xc3, 0xf9, 0x11, 0x58, 0xc4, 0xa8 | RT.SetWakeupTime – SetWakeupTime() disables the wakeup time with **Enable** is FALSE at EFI\_TPL\_CALLBACK. | 1. Call SetWakeupTime() with **Enable** is FALSE.  2. Call GetWakeupTime() to get the wakeup time. The return **Enable** must be FALSE. |

## Virtual Memory Services Test

Reference Document:

*UEFI Specification*, Virtual Memory Services Section.

* Virtual Memory Functions

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| SetVirtualAddressMap | Runtime | Used by an OS loader to convert from physical addressing to virtual addressing. |
| ConvertPointer | Runtime | Used by EFI components to convert internal pointers when switching to virtual addressing. |

No test case is designed to verify these functions in the EFI SCT.

## Misc Runtime Services Test

Reference Document:

*UEFI Specification*, Miscellaneous Runtime Services Section.

* Miscellaneous Runtime Services Functions

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ResetSystem | Runtime | Reset the entire platform. |
| UpdateCapsule | Runtime | Passes capsules to the firmware with both virtual and physical mapping. |
| QueryCapsuleCapabilities | Runtime | Estimate if a capsule or capsules can be updated via UpdateCapsule() |
| GetNextHighMonotonicCount | Runtime | Returns the next high 32 bits of the platform’s monotonic counter. |

### ResetSystem()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.2.4.1.1 | 0x26feed7e, 0x1501, 0x4c0a, 0xae, 0xf3, 0x86, 0xd6, 0x6b, 0xe2, 0xfc, 0xd0 | RT.ResetSystem – ResetSystem() resets the platform with **ResetType** is EfiResetCold at EFI\_TPL\_APPLICATION. | 1. Call ResetSystem() with a **ResetType** value of EfiResetCold. The system should be reset. |
| 5.2.4.1.2 | 0x567f8ee9, 0x4e5e, 0x4278, 0x86, 0x3d, 0xdb, 0xc4, 0xd7, 0x4f, 0x0f, 0xba | RT.ResetSystem – ResetSystem() resets the platform with **ResetType** is **EfiResetCold** at EFI\_TPL\_CALLBACK. | 1. Call ResetSystem() with a **ResetType** value of EfiResetCold. The system should be reset. |
| 5.2.4.1.3 | 0xb7a21919, 0xf358, 0x4a1d, 0x85, 0x26, 0xcc, 0x52, 0x4c, 0x52, 0x94, 0xb2 | RT.ResetSystem – ResetSystem() resets the platform with **ResetType** is EfiResetCold at EFI\_TPL\_NOTIFY. | 1. Call ResetSystem() with a **ResetType** value of EfiResetCold. The system should be reset. |
| 5.2.4.1.4 | 0x7bbad1aa, 0x88b4, 0x4d66, 0x95, 0x94, 0xdb, 0x7e, 0x65, 0xe1, 0xd3, 0xa4 | RT.ResetSystem – ResetSystem() resets the platform with **ResetType** is EfiResetWarm at EFI\_TPL\_APPLICATION. | 1. Call ResetSystem() with a **ResetType** value of EfiResetWarm. The system should be reset. |
| 5.2.4.1.5 | 0xdbe1128b, 0x5155, 0x4241, 0x84, 0x1e, 0x54, 0xea, 0x76, 0x3a, 0x85, 0xc9 | RT.ResetSystem – ResetSystem() resets the platform with **ResetType** is EfiResetWarm at EFI\_TPL\_CALLBACK. | 1. Call ResetSystem() with a **ResetType** value of EfiResetWarm. The system should be reset. |
| 5.2.4.1.6 | 0x8128b536, 0x0b56, 0x480b, 0xa2, 0xd4, 0xcd, 0x79, 0xf8, 0xfa, 0xcb, 0x3b | RT.ResetSystem – ResetSystem() resets the platform with **ResetType** is EfiResetWarm at EFI\_TPL\_NOTIFY. | 1. Call ResetSystem() with a **ResetType** value of EfiResetWarm. The system should be reset. |
| 5.2.4.1.7 | 0x1189a0df, 0xe9cc, 0x45e6, 0xbb, 0x94, 0x21, 0xa7, 0xb3, 0x42, 0x70, 0x96 | RT.ResetSystem – ResetSystem() resets the platform with **ResetType** is EfiResetShutdown at EFI\_TPL\_APPLICATION. | 1. Call ResetSystem() with a **ResetType** value of EfiResetShutdown. The system should be reset or shut down. |
| 5.2.4.1.8 | 0x22b8b295, 0x62a2, 0x4e14, 0xb8, 0x5b, 0xd2, 0xde, 0x36, 0x37, 0x15, 0xb5 | RT.ResetSystem – ResetSystem() resets the platform with **ResetType** is EfiResetShutdown at EFI\_TPL\_CALLBACK. | 1. Call ResetSystem() with a **ResetType** value of EfiResetShutdown. The system should be reset or shut down. |
| 5.2.4.1.9 | 0x1ed1babb, 0x6521, 0x4515, 0x93, 0x9a, 0x39, 0x26, 0xc8, 0xe3, 0x12, 0xff | RT.ResetSystem – ResetSystem() resets the platform with **ResetType** is EfiResetShutdown at EFI\_TPL\_NOTIFY. | 1. Call ResetSystem() with a **ResetType** value of EfiResetShutdown. The system should be reset or shut down. |

### UpdateCapsule()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.2.4.2.1 | 0xf48a2ac4, 0xbce7, 0x4fa7, 0x9e, 0x1b, 0xb9, 0x6f, 0xf8, 0x60, 0xe3, 0x0a | RT.UpdateCapsule –UpdateCapsule() returns EFI\_INVALID\_PARAMETER or EFI\_UNSUPPORTED with **CapsuleCount** is NULL. | 1. Call UpdateCapsule() with a **CapsuleCount** value of NULL. The return value should be EFI\_INVALID\_PARAMETER or EFI\_UNSUPPORTED. |
| 5.2.4.2.2 | 0x304f6960, 0x79d0, 0x4f17, 0x88, 0x11, 0x62, 0x0f, 0xc6, 0xbd, 0xb0, 0xd4 | RT. UpdateCapsule–UpdateCapsule() returns EFI\_INVALID\_PARAMETER or EFI\_UNSUPPORTED when a capsule has the **CAPSULE\_FLAGS\_PERSIST\_ACROSS\_RESET** in its header, but the **ScatterGatherList** is NULL. | 1. Call UpdateCapsule() with **ScatterGatherList** is NULL and a capsule has the flag of **CAPSULE\_FLAGS\_PERSIST\_ACROSS\_RESET** in its header.The return value should be EFI\_INVALID\_PARAMETER or EFI\_UNSUPPORTED. |
| 5.2.4.2.3 | 0x18f86bf8, 0x76cf, 0x4225, 0x8e, 0x3e, 0x1b, 0x1f, 0x63, 0x43, 0x26, 0x00 | RT.UpdateCapsule– UpdateCapsule()returns EFI\_INVALID\_PARAMETER or EFI\_UNSUPPORTED when a capsule has the flag of **CAPSULE\_FLAGS\_POPULATE\_SYSTEM\_TABLE** in its header only. | 1. Call UpdateCapsule()when a capsule has the flag of **CAPSULE\_FLAGS\_POPULATE\_SYSTEM\_TABLE** in its header only. The return value should be EFI\_INVALID\_PARAMETER or EFI\_UNSUPPORTED |

### QueryCapsuleCapabilities()

|  |  |  |  |
| --- | --- | --- | --- |
| Number | GUID | Assertion | Test Description |
| 5.2.4.3.1 | 0x5b5f42d4, 0x8985, 0x45a0, 0x9d, 0xf2, 0x21, 0xaf, 0x74, 0xb1, 0xf5, 0xf6 | RT.QueryCapsuleCapabilities– QueryCapsuleCapabilities() query for generic capsule capability with a fake EFI\_CAPSULE\_HEADER. CAPSULE\_FLAGS\_PERSIST\_ACROSS\_RESET is set in the flags in the header. | 1. Call QueryCapsuleCapabilities() with a fake EFI\_CAPSULE\_HEADER. The return value should be EFI\_SUCCESS or EFI\_UNSUPPORTED. |
| 5.2.4.3.2 | 0x13826168, 0xfef6, 0x407e, 0x93, 0x7c, 0x6d, 0x5e, 0x32, 0x34, 0x9d, 0x5c | RT.QueryCapsuleCapabilities–QueryCapsuleCapabilities() query for generic capsule capability with a fake EFI\_CAPSULE\_HEADER. 0 is set in the flags in the header. | 1. Call QueryCapsuleCapabilities() with a fake EFI\_CAPSULE\_HEADER. The return value should be EFI\_SUCCESS or EFI\_UNSUPPORTED. |
| 5.2.4.3.3 | 0x67c3c36d, 0x4cf8, 0x41fb, 0xa7, 0x8a, 0x86, 0x36, 0x84, 0xe9, 0xe6, 0xe4 | RT.QueryCapsuleCapabilities– QueryCapsuleCapabilities()query for generic capsule capability with **MaximumCapsuleSize** is NULL. | 1. Call QueryCapsuleCapabilities() with **MaximumCapsuleSize** is NULL. The return value should be EFI\_INVALID\_PARAMETER or EFI\_UNSUPPORTED |

### GetNextHighMonotonicCount()

This function may only be available in Runtime. No test case is designed to verify it.