Distribution Package Specification

3/13/2009

Acknowledgements

Revision History

Revision	Revision History	Date
1.0	First release	3/13/2009

Contents

1

Introduction 1

1.1 Overview 1
1.2 Related Information 1
1.3 Definition of Terms 2
1.4 Target Audience 8
1.5 Conventions Used in This Document 8
1.5.1 Table Conventions 9
1.5.2 Pseudo-Code Conventions 14
1.5.3 Typographic Conventions 14

2

Design Discussion 17

2.1 Purpose 17
2.2 Surface Area Description 19
2.2.1 Conditional Surface Area Entries 20
2.2.2 Surface Area Inheritance 21
2.2.3 Source Code Surface Area 22
2.2.4 Informational Surface Area 22
2.2.5 Conditional Descriptions 22
2.2.6 Binary Surface Area 22
2.2.7 Hardware Surface Area 22
2.3 Surface Area Grammar 22
2.4 Distribution 26
2.5 Tracking 27

3

Distribution Package File 29

3.1 Distribution Requirements 29
3.1.1 Creating a Distribution Package File 30
3.1.2 Installing a Distribution Package File 32
3.1.3 Removing a Distribution 33
3.2 Distribution Rules and Definitions 34

4

Distribution Description XML Schema 37

4.1 DistributionPackage.DistributionHeader 37
4.1.1 DistributionHeader:ReadOnly 38
4.1.2 DistributionHeader:RePackage 38
4.1.3 DistributionHeader.Name 39
4.1.4 DistributionHeader.Name:BaseName 39
4.1.5 DistributionHeader.GUID 39
4.1.6 DistributionHeader.GUID:Version 41
4.1.7 DistributionHeader.Vendor 41

4.1.8 DistributionHeader.Date 41
4.1.9 DistributionHeader.Copyright 41
4.1.10 DistributionHeader.License 43
4.1.11 DistributionHeader.Abstract 43
4.1.12 DistributionHeader.Description 43
4.1.13 DistributionHeader.Signature 44
4.1.14 DistributionHeader.XmlSpecification 44

5

Package Surface Area Description 45

5.1 PackageSurfaceArea.Header 45
5.1.1 Header.Name 46
5.1.2 Header.Name:BaseName 46
5.1.3 Header.GUID 48
5.1.4 Header.GUID:Version 48
5.1.5 Header.Copyright 48
5.1.6 Header.License 50
5.1.7 Header.Abstract 50
5.1.8 Header.Description 50
5.1.9 Header.PackagePath 51
5.2 PackageSurfaceArea.ClonedFrom 51
5.2.1 ClonedFrom.GUID 52
5.2.2 ClonedFrom.GUID:Version 52
5.3 PackageSurfaceArea.LibraryClassDeclarations 52
5.3.1 LibraryClassDeclarations.LibraryClass 54
5.3.2 LibraryClassDeclarations.LibraryClass:Keyword 54
5.3.3 LibraryClassDeclarations.LibraryClass:SupArchList 54
5.3.4 LibraryClassDeclarations.LibraryClass:SupModList 56
5.3.5 LibraryClassDeclarations.LibraryClass.HeaderFile 56
5.3.6 LibraryClassDeclarations.LibraryClass.RecommendedInstance 56
5.3.7 LibraryClassDeclarations.LibraryClass.RecommendedInstance.GUID 58
5.3.8 LibraryClassDeclarations.LibraryClass.RecommendedInstance.GUID:Version . 58
5.3.9 LibraryClassDeclarations.LibraryClass.HelpText 58
5.3.10 LibraryClassDeclarations.LibraryClass.HelpText:Lang 59
5.4 PackageSurfaceArea.IndustryStandardIncludes 59
5.4.1 IndustryStandardIncludes.IndustryStandardHeader 60
5.4.2 IndustryStandardIncludes.IndustryStandardHeader.HeaderFile 60
5.4.3 IndustryStandardIncludes.IndustryStandardHeader.HelpText 60
5.4.4 IndustryStandardIncludes.IndustryStandardHeader.HelpText:Lang 61
5.5 PackageSurfaceArea.PackageIncludes 61
5.5.1 PackageIncludes.PackageHeader 62
5.5.2 PackageIncludes.PackageHeader.HeaderFile 62
5.5.3 PackageIncludes.PackageHeader.HeaderFile:SupArchList 64
5.5.4 PackageIncludes.PackageHeader.HeaderFile:SupModList 64
5.5.5 PackageIncludes.PackageHeader.HelpText 64
5.5.6 PackageIncludes.PackageHeader.HelpText:Lang 65
5.6 PackageSurfaceArea.Modules 65

5.6.1 Modules.ModuleSurfaceArea 65
5.7 PackageSurfaceArea.GuidDeclarations 65
5.7.1 GuidDeclarations.Entry 66
5.7.2 GuidDeclarations.Entry:UiName 67
5.7.3 GuidDeclarations.Entry:GuidTypes 67
5.7.4 GuidDeclarations.Entry:SupArchList 67
5.7.5 GuidDeclarations.Entry:SupModList 68
5.7.6 GuidDeclarations.Entry.CName 68
5.7.7 GuidDeclarations.Entry.GuidValue 68
5.7.8 GuidDeclarations.Entry.HelpText 69
5.7.9 GuidDeclarations.Entry.HelpText:Lang 69
5.8 PackageSurfaceArea.ProtocolDeclarations 69
5.8.1 ProtocolDeclaration.Entry 70
5.8.2 ProtocolDeclarations.Entry:UiName 70
5.8.3 ProtocolDeclarations.Entry:SupArchList 71
5.8.4 ProtocolDeclarations.Entry:SupModList 71
5.8.5 ProtocolDeclarations.Entry.CName 71
5.8.6 ProtocolDeclarations.Entry.GuidValue 73
5.8.7 ProtocolDeclarations.Entry.HelpText 73
5.8.8 ProtocolDeclarations.Entry.HelpText:Lang 73
5.9 PackageSurfaceArea.PpiDeclarations 73
5.9.1 PpiDeclarations.Entry 75
5.9.2 PpiDeclarations.Entry:UiName 75
5.9.3 PpiDeclarations.Entry:SupArchList 76
5.9.4 PpiDeclarations.Entry:SupModList 76
5.9.5 PpiDeclarations.Entry.CName 76
5.9.6 PpiDeclarations.Entry.GuidValue 78
5.9.7 PpiDeclarations.Entry.HelpText 78
5.9.8 PpiDeclarations.Entry.HelpText:Lang 78
5.10 PackageSurfaceArea.PcdDeclarations 78
5.10.1 PcdDeclarations.PcdEntry 79
5.10.2 PcdDeclarations.PcdEntry:SupArchList 79
5.10.3 PcdDeclarations.PcdEntry:SupModList 80
5.10.4 PcdDeclarations.PcdEntry.TokenSpaceGuidCname 80
5.10.5 PcdDeclarations.PcdEntry.Token 82
5.10.6 PcdDeclarations.PcdEntry.CName 82
5.10.7 PcdDeclarations.PcdEntry.DatumType 82
5.10.8 PcdDeclarations.PcdEntry.ValidUsage 82
5.10.9 PcdDeclarations.PcdEntry.DefaultValue 84
5.10.10 PcdDeclarations.PcdEntry.MaxDatumSize 84
5.10.11 PcdDeclarations.PcdEntry.HelpText 84
5.10.12 PcdDeclarations.PcdEntry.HelpText:Lang 86
5.10.13 PcdDeclarations.PcdEntry.PcdError 86
5.10.14 PcdDeclarations.PcdEntry.PcdError.ValidValueList 86
5.10.15 PcdDeclarations.PcdEntry.PcdError.ValidValueList:Lang 88
5.10.16 PcdDeclarations.PcdEntry.PcdError.ValidValueRange 88
5.10.17 PcdDeclarations.PcdEntry.PcdError.Expression 88

5.10.18 PcdDeclarations.PcdEntry.PcdError.ErrorNumber 90
5.10.19 PcdDeclarations.PcdEntry.PcdError.ErrorMessage 90
5.10.20 PcdDeclarations.PcdEntry.PcdError.ErrorMessage:Lang 90
5.11 PackageSurfaceArea.PcdRelationshipChecks 90
5.11.1 PcdRelationshipChecks.PcdCheck 91
5.12 PackageSurfaceArea.MiscellaneousFiles 91
5.12.1 MiscellaneousFiles.Copyright 92
5.12.2 MiscellaneousFiles.License 92
5.12.3 MiscellaneousFiles.Abstract 94
5.12.4 MiscellaneousFiles.Description 94
5.12.5 MiscellaneousFiles.Filename 94
5.12.6 MiscellaneousFiles.Filename:Executable 94
5.13 PackageSurfaceArea.UserExtensions 95
5.13.1 UserExtensions:UserId 96
5.13.2 UserExtensions:Identifier 96

6 Module Surface Area Description 97

6.1 ModuleSurfaceArea:BinaryModule 98
6.2 ModuleSurfaceArea.Header 98
6.2.1 Header.Name 99
6.2.2 Header.Name:BaseName 99
6.2.3 Header.GUID 101
6.2.4 Header.GUID:Version 101
6.2.5 Header.Copyright 101
6.2.6 Header.License 103
6.2.7 Header.Abstract 103
6.2.8 Header.Description 103
6.3 ModuleSurfaceArea.ModuleProperties 104
6.3.1 ModuleProperties:SupArchList 105
6.3.2 ModuleProperties:SupModList 107
6.3.3 ModuleProperties.ModuleType 107
6.3.4 ModuleProperties.Path 107
6.3.5 ModuleProperties.PcdIsDriver 109
6.3.6 ModuleProperties.UefiSpecificationVersion 109
6.3.7 ModuleProperties.PiSpecificationVersion 109
6.3.8 ModuleProperties.Specification 109
6.3.9 ModuleProperties.Specification:Version 111
6.3.10 ModuleProperties.BootMode 111
6.3.11 ModuleProperties.BootMode:Usage 111
6.3.12 ModuleProperties.BootMode:SupArchList 113
6.3.13 ModuleProperties.BootMode:FeatureFlag 113
6.3.14 ModuleProperties.BootMode.SupportedBootModes 113
6.3.15 ModuleProperties.BootMode.HelpText 115
6.3.16 ModuleProperties.BootMode.HelpText:Lang 115
6.3.17 ModuleProperties.Event 115
6.3.18 ModuleProperties.Event:Usage 115

6.3.19 ModuleProperties.Event:EventType 116
6.3.20 ModuleProperties.Event:SupArchList 116
6.3.21 ModuleProperties.Event:FeatureFlag 116
6.3.22 ModuleProperties.Event.HelpText 117
6.3.23 ModuleProperties.Event.HelpText:Lang 117
6.3.24 ModuleProperties.HOB 119
6.3.25 ModuleProperties.HOB:Usage 119
6.3.26 ModuleProperties.HOB:HobType 119
6.3.27 ModuleProperties.HOB:SupArchList 120
6.3.28 ModuleProperties.HOB:FeatureFlag 121
6.3.29 ModuleProperties.HOB.HelpText 121
6.3.30 ModuleProperties.HOB.HelpText:Lang 121
6.4 ModuleSurfaceArea.ClonedFrom 122
6.4.1 ModuleSurfaceArea.ClonedFrom 123
6.4.2 ClonedFrom.GUID 123
6.4.3 ClonedFrom.GuidValue:Version 123
6.5 ModuleSurfaceArea.LibraryClassDefinitions 123
6.5.1 LibraryClassDefinitions.LibraryClass 124
6.5.2 LibraryClassDefinitions.LibraryClass:Usage 124
6.5.3 LibraryClassDefinitions.LibraryClass:SupArchList 126
6.5.4 LibraryClassDefinitions.LibraryClass:SupModList 126
6.5.5 LibraryClassDefinitions.LibraryClass:FeatureFlag 126
6.5.6 LibraryClassDefinitions.LibraryClass.RecommendedInstance 127
6.5.7 LibraryClassDefinitions.LibraryClass.RecommendedInstance.GUID 127
6.5.8 LibraryClassDefinitions.LibraryClass.RecommendedInstance.GUID:Version .. 128
6.5.9 LibraryClassDefinitions.LibraryClass.HelpText 128
6.5.10 LibraryClassDefinitions.LibraryClass.HelpText:Lang 128
6.6 ModuleSurfaceArea.SourceFiles 129
6.6.1 SourceFiles.Filename 130
6.6.2 SourceFiles.Filename:Family 130
6.6.3 SourceFiles.Filename:SupArchList 131
6.6.4 SourceFiles.Filename:FeatureFlag 131
6.7 ModuleSurfaceArea.BinaryFiles 131
6.7.1 BinaryFiles.BinaryFile 132
6.7.2 BinaryFiles.BinaryFile.Filename 132
6.7.3 BinaryFiles.BinaryFile.Filename:FileType 134
6.7.4 BinaryFiles.BinaryFile.Filename:SupArchList 134
6.7.5 BinaryFiles.BinaryFile.Filename:FeatureFlag 134
6.7.6 BinaryFiles.BinaryFile.AsBuilt 135
6.7.7 BinaryFiles.BinaryFile.AsBuilt.PatchPcdValue 136
6.7.8 BinaryFiles.BinaryFile.AsBuilt.PatchPcdValue.TokenSpaceGuidValue 136
6.7.9 BinaryFiles.BinaryFile.AsBuilt.PatchPcdValue.Token 137
6.7.10 BinaryFiles.BinaryFile.AsBuilt.PatchPcdValue.PcdCName 137
6.7.11 BinaryFiles.BinaryFile.AsBuilt.PatchPcdValue.DatumType 137
6.7.12 BinaryFiles.BinaryFile.AsBuilt.PatchPcdValue.MaxDatumSize 137
6.7.13 BinaryFiles.BinaryFile.AsBuilt.PatchPcdValue.Value 138
6.7.14 BinaryFiles.BinaryFile.AsBuilt.PatchPcdValue.Offset 138

6.7.15 BinaryFiles.BinaryFile.AsBuilt.PatchPcdValue.HelpText 138
6.7.16 BinaryFiles.BinaryFile.AsBuilt.PatchPcdValue.HelpText:Lang 140
6.7.17 BinaryFiles.BinaryFile.AsBuilt.PatchPcdValue.PcdError 140
6.7.18 BinaryFiles.BinaryFile.AsBuilt.PatchPcdValue.PcdError.ErrorNumber..... 140
6.7.19 BinaryFiles.BinaryFile.AsBuilt.PatchPcdValue.PcdError.ErrorMessage..... 141
6.7.20 BinaryFiles.BinaryFile.AsBuilt.PatchPcdValue.PcdError.ErrorMessage:Lang 141
6.7.21 BinaryFiles.BinaryFile.AsBuilt.PcdExValue..... 141
6.7.22 BinaryFiles.BinaryFile.AsBuilt.PcdExValue.TokenSpaceGuidValue..... 143
6.7.23 BinaryFiles.BinaryFile.AsBuilt.PcdExValue.Token..... 143
6.7.24 BinaryFiles.BinaryFile.AsBuilt.PcdExValue.DatumType 143
6.7.25 BinaryFiles.BinaryFile.AsBuilt.PcdExValue.MaxDatumSize 143
6.7.26 BinaryFiles.BinaryFile.AsBuilt.PcdExValue.Value 145
6.7.27 BinaryFiles.BinaryFile.AsBuilt.PcdExValue.HelpText 145
6.7.28 BinaryFiles.BinaryFile.AsBuilt.PcdExValue.HelpText:Lang 145
6.7.29 BinaryFiles.BinaryFile.AsBuilt.PcdExValue.PcdError 145
6.7.30 BinaryFiles.BinaryFile.AsBuilt.PcdExValue.PcdError.ErrorNumber..... 146
6.7.31 BinaryFiles.BinaryFile.AsBuilt.PcdExValue.PcdError.ErrorMessage..... 146
6.7.32 BinaryFiles.BinaryFile.AsBuilt.PcdExValue.PcdError.ErrorMessage:Lang..... 146
6.7.33 BinaryFiles.BinaryFile.AsBuilt.LibraryInstances..... 148
6.7.34 BinaryFiles.BinaryFile.AsBuilt.LibraryInstances.GUID..... 148
6.7.35 BinaryFiles.BinaryFile.AsBuilt.LibraryInstances.GUID:Version..... 148
6.7.36 BinaryFiles.BinaryFile.AsBuilt.BuildFlags 148
6.8 ModuleSurfaceArea.PackageDependencies 149
6.8.1 PackageDependencies.Package 150
6.8.2 PackageDependencies.Package:SupArchList..... 150
6.8.3 PackageDependencies.Package:FeatureFlag..... 152
6.8.4 PackageDependencies.Package.Description 152
6.8.5 PackageDependencies.Package.GUID 152
6.8.6 PackageDependencies.Package.GUID:Version 152
6.9 ModuleSurfaceArea.Guids..... 153
6.9.1 Guids.GuidCName..... 154
6.9.2 Guids.GuidCName..... 154
6.9.3 Guids.GuidCName:Usage..... 155
6.9.4 Guids.GuidCName:GuidType 156
6.9.5 Guids.GuidCName:SupArchList..... 158
6.9.6 Guids.GuidCName:FeatureFlag..... 158
6.9.7 Guids.GuidCName.CName..... 158
6.9.8 Guids.GuidCName.VariableName 160
6.9.9 Guids.GuidCName.HelpText..... 160
6.9.10 Guids.GuidCName.HelpText:Lang..... 160
6.10 ModuleSurfaceArea.Protocols 160
6.10.1 Protocols.Protocol 161
6.10.2 Protocols.Protocol:Usage..... 161
6.10.3 Protocols.Protocol:Notify..... 163
6.10.4 Protocols.Protocol:SupArchList..... 163
6.10.5 Protocols.Protocol:FeatureFlag..... 163
6.10.6 Protocols.Protocol.CName..... 165

6.10.7 Protocols.Protocol.HelpText 165
6.10.8 Protocols.Protocol.HelpText:Lang 165
6.11 ModuleSurfaceArea.PPIs 165
6.11.1 PPIs.Ppi 167
6.11.2 PPIs.Ppi:Usage 167
6.11.3 PPIs.Ppi:Notify 169
6.11.4 PPIs.Ppi:SupArchList 169
6.11.5 PPIs.Ppi:FeatureFlag 169
6.11.6 PPIs.Ppi.CName 171
6.11.7 PPIs.Ppi.HelpText 171
6.11.8 PPIs.Ppi.HelpText:Lang 171
6.12 ModuleSurfaceArea.Externs 171
6.12.1 Externs.Extern 172
6.12.2 Externs.Extern:SupArchList 172
6.12.3 Externs.Extern:FeatureFlag 173
6.12.4 Externs.Extern.EntryPoint 173
6.12.5 Externs.Extern.UnloadImage 175
6.12.6 Externs.Extern.Constructor 175
6.12.7 Externs.Extern.Destructor 175
6.12.8 Externs.Extern.HelpText 175
6.12.9 Externs.Extern.HelpText:Lang 176
6.13 ModuleSurfaceArea.PcdCoded 176
6.13.1 PcdCoded.PcdEntry 177
6.13.2 PcdCoded.PcdEntry:PcdItemType 177
6.13.3 PcdCoded.PcdEntry:PcdUsage 178
6.13.4 PcdCoded.PcdEntry:SupArchList 180
6.13.5 PcdCoded.PcdEntry:FeatureFlag 180
6.13.6 PcdCoded.PcdEntry.CName 180
6.13.7 PcdCoded.PcdEntry.TokenSpaceGuidCName 182
6.13.8 PcdCoded.PcdEntry.DefaultValue 182
6.13.9 PcdCoded.PcdEntry.HelpText 182
6.13.10 PcdCoded.PcdEntry.HelpText:Lang 183
6.14 ModuleSurfaceArea.PeiDepex 183
6.14.1 PeiDepex.Expression 184
6.14.2 PeiDepex.HelpText 185
6.14.3 PeiDepex.HelpText:Lang 185
6.15 Module SurfaceArea.DxeDepex 185
6.15.1 DxeDepex.Expression 186
6.15.2 DxeDepex.HelpText 187
6.15.3 DxeDepex.HelpText:Lang 187
6.16 Module SurfaceArea.SmmDepex 187
6.16.1 SmmDepex.Expression 188
6.16.2 SmmDepex.HelpText 189
6.16.3 SmmDepex.HelpText:Lang 189
6.17 ModuleSurfaceArea.MiscellaneousFiles 189
6.17.1 MiscellaneousFiles.Description 191
6.17.2 MiscellaneousFiles.Filename 191

6.17.3 Misc.Filename:Executable 191
6.18 ModuleSurfaceArea.UserExtensions 192
6.18.1 UserExtensions:UserId 193
6.18.2 UserExtensions:Identifier 193

7

Tools Surface Area Description 195

7.1 Tools.Header 195
7.1.1 Header.Name 196
7.1.2 Header.Copyright 196
7.1.3 Header.License 196
7.1.4 Header.Abstract 198
7.1.5 Header.Description 198
7.2 Tools.Filename 198
7.2.1 Filename:OS 199
7.2.2 Filename:Executable 199

8

MiscellaneousFiles Section 201

8.1 MiscellaneousFiles.Header 201
8.1.1 Header.Name 202
8.1.2 Header.Copyright 202
8.1.3 Header.License 202
8.1.4 Header.Abstract 203
8.1.5 Header.Description 203
8.2 MiscellaneousFiles.Filename 203
8.2.1 Filename:Executable 204

9

UserExtensions 205

9.1 UserExtensions:UserId 206
9.2 UserExtensions:Identifier 206

Appendix A

XML Examples 207

A.1Distribution Package Header 207
A.2Package Header 208
A.3UEFI Application 210
A.4UEFI Application Source Files 213
A.5UEFI Driver 213
A.6Library 215
A.7PI PEIM 217
A.8PI DXE Driver 219
A.9Tool 221

Figures

Figure 1. Module Surface Area Differs for Different Binaries 21
Figure 2. Distribution Package File layout 30
Figure 3. NoSuchCorpPkg Directory Tree 31

Tables

Table 1. Module Type Names 4
Table 2. Base XML Data Types 9
Table 3. Value Types 9
Table 4. Enumeration Types 10
Table 5. List Types 13
Table 6. Table Description 14
Table 7. Usage Description 23
Table 8. Surface Area Grammar Summary 24
Table 9. Distribution Rules & Definitions KEY 34
Table 10. DistributionPackage 37
Table 11. DistributionPackage.DistributionHeader 38
Table 12. DistributionHeader:ReadOnly 38
Table 13. DistributionHeader:RePackage 39
Table 14. DistributionHeader.Name 39
Table 15. DistributionHeader.Name:BaseName 39
Table 16. DistributionHeader.GUID 39
Table 17. DistributionHeader.GUID:Version 41
Table 18. DistributionHeader.Vendor 41
Table 19. DistributionHeader.Date 41
Table 20. DistributionHeader.Copyright 41
Table 21. DistributionHeader.License 43
Table 22. DistributionHeader.Abstract 43
Table 23. DistributionHeader.Description 44
Table 24. DistributionHeader.Signature 44
Table 25. DistributionHeader.XmlSpecification 44
Table 26. DistributionPackage.PackageSurfaceArea 45
Table 27. PackageSurfaceArea.Header 46
Table 28. Header.Name 46
Table 29. Header.Name:BaseName 46
Table 30. Header.GUID 48
Table 31. Header.GUID:Version 48
Table 32. Header.Copyright 48
Table 33. Header.License 50
Table 34. Header.Abstract 50
Table 35. Header.Description 51
Table 36. Header.PackagePath 51
Table 37. PackageSurfaceArea.ClonedFrom 52
Table 38. ClonedFrom.GUID 52
Table 39. ClonedFrom.GUID:Version 52
Table 40. PackageSurfaceArea.LibraryClassDeclarations 54
Table 41. LibraryClassDeclarations.LibraryClass 54
Table 42. LibraryClassDeclarations.LibraryClass:Keyword 54
Table 43. LibraryClassDeclarations.LibraryClass:SupArchList 54

Table 44. LibraryClassDeclarations.LibraryClass:SupModList 56
Table 45. LibraryClassDeclarations.LibraryClass.HeaderFile 56
Table 46. LibraryClassDeclarations.LibraryClass.RecommendedInstance 56
Table 47. LibraryClassDeclarations.LibraryClass.RecommendedInstance.GUID 58
Table 48. LibraryClassDeclarations.LibraryClass:RecommendedInstance.GUID:Version ... 58
Table 49. LibraryClassDeclarations.LibraryClass.HelpText 58
Table 50. LibraryClassDeclarations.LibraryClass.HelpText:Lang 59
Table 51. PackageSurfaceArea.IndustryStandardIncludes 60
Table 52. IndustryStandardIncludes.IndustryStandardHeader 60
Table 53. IndustryStandardIncludes.IndustryStandardHeader.HeaderFile 60
Table 54. IndustryStandardIncludes.IndustryStandardHeader.HelpText 61
Table 55. IndustryStandardIncludes.IndustryStandardHeader.HelpText:Lang 61
Table 56. PackageSurfaceArea.PackageIncludes 62
Table 57. PackageIncludes.PackageHeader 62
Table 58. PackageIncludes.PackageHeader.HeaderFile 62
Table 59. PackageIncludes.PackageHeader.HeaderFile:SupArchList 64
Table 60. PackageIncludes.PackageHeader.HeaderFile:SupModList 64
Table 61. PackageIncludes.PackageHeader.HelpText 64
Table 62. PackageIncludes.PackageHeader.HelpText:Lang 65
Table 63. PackageSurfaceArea.Modules 65
Table 64. PackageSurfaceArea.GuidDeclarations 66
Table 65. GuidDeclarations.Entry 66
Table 66. GuidDeclarations.Entry:UiName 67
Table 67. GuidDeclarations.Entry:GuidTypes 67
Table 68. GuidDeclarations.Entry:SupArchList 67
Table 69. GuidDeclarations.Entry:SupModList 68
Table 70. GuidDeclarations.Entry.CName 68
Table 71. GuidDeclarations.Entry.GuidValue 68
Table 72. GuidDeclarations.Entry.HelpText 69
Table 73. GuidDeclarations.Entry.HelpText:Lang 69
Table 74. PackageSurfaceArea.ProtocolDeclarations 70
Table 75. ProtocolDeclarations.Entry 70
Table 76. ProtocolDeclarations.Entry:UiName 70
Table 77. ProtocolDeclarations.Entry:SupArchList 71
Table 78. ProtocolDeclarations.Entry:SupModList 71
Table 79. ProtocolDeclarations.Entry.CName 71
Table 80. ProtocolDeclarations.Entry.GuidValue 73
Table 81. ProtocolDeclarations.Entry.HelpText 73
Table 82. ProtocolDeclarations.Entry.HelpText:Lang 73
Table 83. PackageSurfaceArea.PpiDeclarations 75
Table 84. PpiDeclarations.Entry 75
Table 85. PpiDeclarations.Entry:UiName 75
Table 86. PpiDeclarations.Entry:SupArchList 76
Table 87. PpiDeclarations.Entry:SupModList 76
Table 88. PpiDeclarations.Entry.CName 76
Table 89. PpiDeclarations.Entry.GuidValue 78
Table 90. PpiDeclarations.Entry.HelpText 78

Table 91. PpiDeclarations.Entry.HelpText:Lang 78
Table 92. PackageSurfaceArea.PcdDeclarations 79
Table 93. PcdDeclarations.PcdEntry 79
Table 94. PcdDeclarations.PcdEntry:SupArchList 80
Table 95. PcdDeclarations.PcdEntry:SupModList 80
Table 96. PcdDeclarations.PcdEntry.TokenSpaceGuidCname 80
Table 97. PcdDeclarations.PcdEntry.Token 82
Table 98. PcdDeclarations.PcdEntry.CName 82
Table 99. PcdDeclarations.PcdEntry.DatumType 82
Table 100. PcdDeclarations.PcdEntry.ValidUsage 82
Table 101. PcdDeclarations.PcdEntry.DefaultValue 84
Table 102. PcdDeclarations.PcdEntry.MaxDatumSize 84
Table 103. PcdDeclarations.PcdEntry.HelpText 84
Table 104. PcdDeclarations.PcdEntry.HelpText:Lang 86
Table 105. PcdDeclarations.PcdEntry.PcdError 86
Table 106. PcdDeclarations.PcdEntry.PcdError.ValidValueList 86
Table 107. PcdDeclarations.PcdEntry.PcdError.ValidValueList:Lang 88
Table 108. PcdDeclarations.PcdEntry.PcdError.ValidValueRange 88
Table 109. PcdDeclarations.PcdEntry.PcdError.Expression 88
Table 110. PcdDeclarations.PcdEntry.PcdError.ErrorNumber 90
Table 111. PcdDeclarations.PcdEntry.PcdError.ErrorMessage 90
Table 112. PcdDeclarations.PcdEntry.PcdError.ErrorMessage:Lang 90
Table 113. PackageSurfaceArea.PcdRelationshipChecks 91
Table 114. PcdRelationshipChecks.PcdCheck 91
Table 115. PackageSurfaceArea.MiscellaneousFiles 92
Table 116. MiscellaneousFiles.Copyright 92
Table 117. MiscellaneousFiles.License 92
Table 118. MiscellaneousFiles.Abstract 94
Table 119. MiscellaneousFiles.Description 94
Table 120. MiscellaneousFiles.Filename 94
Table 121. MiscellaneousFiles.Filename:Executable 94
Table 122. PackageSurfaceArea.UserExtensions 96
Table 123. UserExtensions:UserId 96
Table 124. UserExtensions:Identifier 96
Table 125. ModuleSurfaceArea 98
Table 126. ModuleSurfaceArea:BinaryModule 98
Table 127. ModuleSurfaceArea.Header 99
Table 128. Header.Name 99
Table 129. Header.Name:BaseName 99
Table 130. Header.GUID 101
Table 131. Header.GUID:Version 101
Table 132. Header.Copyright 101
Table 133. Header.License 103
Table 134. Header.Abstract 103
Table 135. Header.Description 104
Table 136. ModuleSurfaceArea.ModuleProperties 105
Table 137. ModuleProperties:SupArchList 105

Table 138. ModuleProperties:SupModList 107
Table 139. ModuleProperties.ModuleType 107
Table 140. ModuleProperties.Path 107
Table 141. ModuleProperties.PcdIsDriver 109
Table 142. ModuleProperties.UefiSpecificationVersion 109
Table 143. ModuleProperties.PiSpecificationVersion 109
Table 144. ModuleProperties.Specification 109
Table 145. ModuleProperties.Specification:Version 111
Table 146. ModuleProperties.BootMode 111
Table 147. ModuleProperties.BootMode:Usage 111
Table 148. ModuleProperties.BootMode:SupArchList 113
Table 149. ModuleProperties.BootMode:FeatureFlag 113
Table 150. ModuleProperties.BootMode.SupportedBootMode 113
Table 151. ModuleProperties.BootMode.HelpText 115
Table 152. ModuleProperties.BootMode.HelpText:Lang 115
Table 153. ModuleSurfaceArea.ModuleProperties.Event 115
Table 154. ModuleProperties.Event:Usage 116
Table 155. ModuleProperties.Event:EventType 116
Table 156. ModuleProperties.Event:SupArchList 116
Table 157. ModuleProperties.Event:FeatureFlag 117
Table 158. ModuleProperties.Event.HelpText 117
Table 159. ModuleProperties.Event.HelpText:Lang 117
Table 160. ModuleSurfaceArea.ModuleProperties.HOB 119
Table 161. ModuleProperties.HOB:Usage 119
Table 162. ModuleProperties.HOB:HobType 119
Table 163. ModuleProperties.HOB:SupArchList 120
Table 164. ModuleProperties.HOB:FeatureFlag 121
Table 165. ModuleProperties.HOB.HelpText 121
Table 166. ModuleProperties.HOB.HelpText:Lang 121
Table 167. ModuleSurfaceArea.ClonedFrom 123
Table 168. ClonedFrom.GUID 123
Table 169. ClonedFrom.GuidValue:Version 123
Table 170. ModuleSurfaceArea.LibraryClassDefinitions 124
Table 171. LibraryClassDefinitions.LibraryClass 124
Table 172. LibraryClassDefinitions.LibraryClass:Usage 124
Table 173. LibraryClassDefinitions.LibraryClass:SupArchList 126
Table 174. LibraryClassDefinitions.LibraryClass:SupModList 126
Table 175. LibraryClassDefinitions.LibraryClass:FeatureFlag 127
Table 176. LibraryClassDefinitions.LibraryClass.Keyword 127
Table 177. LibraryClassDefinitions.LibraryClass.RecommendedInstance 127
Table 178. LibraryClassDefinitions.LibraryClass.RecommendedInstance.GUID 128
Table 179. LibraryClassDefinitions.LibraryClass:RecommendedInstance.GUID:Version .. 128
Table 180. LibraryClassDefinitions.LibraryClass.HelpText 128
Table 181. LibraryClassDefinitions.LibraryClass.HelpText:Lang 128
Table 182. ModuleSurfaceArea.SourceFiles 130
Table 183. SourceFiles.Filename 130
Table 184. SourceFiles.Filename:Family 130

Table 185. SourceFiles.Filename:SupArchList 131
Table 186. SourceFiles.Filename:FeatureFlag 131
Table 187. ModuleSurfaceArea.BinaryFiles 132
Table 188. BinaryFiles.BinaryFile 132
Table 189. BinaryFiles.BinaryFile.Filename 132
Table 190. BinaryFiles.BinaryFile.Filename:FileType 134
Table 191. BinaryFiles.BinaryFile.Filename:SupArchList 134
Table 192. BinaryFiles.BinaryFile.Filename:FeatureFlag 134
Table 193. BinaryFiles.BinaryFile.AsBuilt 136
Table 194. BinaryFiles.BinaryFile.AsBuilt.PatchPcdValue 136
Table 195. BinaryFiles.BinaryFile.AsBuilt.PatchPcdValue.TokenSpaceGuidValue 137
Table 196. BinaryFiles.BinaryFile.AsBuilt.PatchPcdValue.Token 137
Table 197. BinaryFiles.BinaryFile.AsBuilt.PatchPcdValue.PcdCName 137
Table 198. BinaryFiles.BinaryFile.AsBuilt.PatchPcdValue.DatumType 137
Table 199. BinaryFiles.BinaryFile.AsBuilt.PatchPcdValue.MaxDatumSize 138
Table 200. BinaryFiles.BinaryFile.AsBuilt.PatchPcdValue.Value 138
Table 201. BinaryFiles.BinaryFile.AsBuilt.PatchPcdValue.Offset 138
Table 202. BinaryFiles.BinaryFile.AsBuilt.PatchPcdValue.HelpText 138
Table 203. BinaryFiles.BinaryFile.AsBuilt.PatchPcdValue.HelpText:Lang 140
Table 204. BinaryFiles.BinaryFile.AsBuilt.PatchPcdValue.PcdError 140
Table 205. BinaryFiles.BinaryFile.AsBuilt.PatchPcdValue.PcdError.ErrorNumber 140
Table 206. BinaryFiles.BinaryFile.AsBuilt.PatchPcdValue.PcdError.ErrorMessage 141
Table 207. BinaryFiles.BinaryFile.AsBuilt.PatchPcdValue.PcdError.ErrorMessage:Lang .. 141
Table 208. BinaryFiles.BinaryFile.AsBuilt.PcdExValue 141
Table 209. BinaryFiles.BinaryFile.AsBuilt.PcdExValue.TokenSpaceGuidValue 143
Table 210. BinaryFiles.BinaryFile.AsBuilt.PcdExValue.Token 143
Table 211. BinaryFiles.BinaryFile.AsBuilt.PcdExValue.DatumType 143
Table 212. BinaryFiles.BinaryFile.AsBuilt.PcdExValue.MaxDatumSize 143
Table 213. BinaryFiles.BinaryFile.AsBuilt.PcdExValue.Value 145
Table 214. BinaryFiles.BinaryFile.AsBuilt.PcdExValue.HelpText 145
Table 215. BinaryFiles.BinaryFile.AsBuilt.PcdExValue.HelpText:Lang 145
Table 216. BinaryFiles.BinaryFile.AsBuilt.PcdExValue.PcdError 146
Table 217. BinaryFiles.BinaryFile.AsBuilt.PcdExValue.PcdError.ErrorNumber 146
Table 218. BinaryFiles.BinaryFile.AsBuilt.PcdExValue.PcdError.ErrorMessage 146
Table 219. BinaryFiles.BinaryFile.AsBuilt.PcdExValue.PcdError.ErrorMessage:Lang 146
Table 220. BinaryFiles.BinaryFile.AsBuilt.LibraryInstances 148
Table 221. BinaryFiles.BinaryFile.AsBuilt.LibraryInstances:GUID 148
Table 222. BinaryFiles.BinaryFile.AsBuilt.LibraryInstances:Version 148
Table 223. BinaryFiles.BinaryFile.AsBuilt.BuildFlags 149
Table 224. ModuleSurfaceArea.PackageDependencies 150
Table 225. PackageDependencies.Package 150
Table 226. PackageDependencies.Package:SupArchList 150
Table 227. PackageDependencies.Package:FeatureFlag 152
Table 228. PackageDependencies.Package.Description 152
Table 229. PackageDependencies.Package.GUID 152
Table 230. PackageDependencies.Package.GUID:Version 153
Table 231. ModuleSurfaceArea.Guids 154

Table 232. Guids.GuidCName 154
Table 233. Guids.GuidCName 154
Table 234. Guids.GuidCName:Usage 155
Table 235. Guids.GuidCName:GuidType 156
Table 236. Guids.GuidCName:SupArchList 158
Table 237. Guids.GuidCName:FeatureFlag 158
Table 238. Guids.GuidCName.CName 158
Table 239. Guids.GuidCName.VariableName 160
Table 240. Guids.GuidCName.HelpText 160
Table 241. Guids.GuidCName.HelpText:Lang 160
Table 242. ModuleSurfaceArea.Protocols 161
Table 243. Protocols.Protocol 161
Table 244. Protocols.Protocol:Usage 161
Table 245. Protocols.Protocol:Notify 163
Table 246. Protocols.Protocol:SupArchList 163
Table 247. Protocols.Protocol:FeatureFlag 163
Table 248. Protocols.Protocols.CName 165
Table 249. Protocols.Protocol.HelpTex 165
Table 250. Protocols.Protocol.HelpText:Lang 165
Table 251. ModuleSurfaceArea.PPIs 167
Table 252. PPIs.Ppi 167
Table 253. PPIs.Ppi:Usage 167
Table 254. Ppis.Ppi:Notify 169
Table 255. PPIs.Ppi:SupArchList 169
Table 256. PPIs.Ppi:FeatureFlag 169
Table 257. PPIs.Ppi.CName 171
Table 258. PPIs.Ppi.HelpTex 171
Table 259. PPIs.Ppi.HelpText:Lang 171
Table 260. ModuleSurfaceArea.Externs 172
Table 261. Externs.Extern 172
Table 262. Externs.Extern:SupArchList 173
Table 263. Externs.Extern:FeatureFlag 173
Table 264. Externs.Extern.EntryPoint 173
Table 265. Externs.Extern.UnloadImage 175
Table 266. Externs.Extern.Constructor 175
Table 267. Externs.Extern.Destructor 175
Table 268. Externs.Extern.HelpText 176
Table 269. Externs.Extern.HelpText:Lang 176
Table 270. ModuleSurfaceArea.PcdCoded 177
Table 271. ModuleSurfaceArea.PcdCoded.PcdEntry 177
Table 272. PcdCoded.PcdEntry:PcdItemType 177
Table 273. PcdCoded.PcdEntry:PcdUsage 178
Table 274. PcdCoded.PcdEntry:SupArchList 180
Table 275. PcdCoded.PcdEntry:FeatureFlag 180
Table 276. PcdCoded.PcdEntry.CName 180
Table 277. PcdCoded.PcdEntry.TokenSpaceGuidCName 182
Table 278. PcdCoded.PcdEntry.DefaultValue 182

Table 279. PcdCoded.PcdEntry.HelpText 182
Table 280. PcdCoded.PcdEntry.HelpText:Lang 183
Table 281. ModuleSurfaceArea.PeiDepex 184
Table 282. PeiDepex.Expression 184
Table 283. PeiDepex.HelpText 185
Table 284. PeiDepex.HelpText:Lang 185
Table 285. ModuleSurfaceArea.DxeDepex 186
Table 286. DxeDepex.Expression 186
Table 287. DxeDepex.HelpText 187
Table 288. DxeDepex.HelpText:Lang 187
Table 289. ModuleSurfaceArea.SmmDepex 188
Table 290. SmmDepex.Expression 188
Table 291. SmmDepex.HelpText 189
Table 292. SmmDepex.HelpText:Lang 189
Table 293. ModuleSurfaceArea.MiscellaneousFiles 191
Table 294. MiscellaneousFiles.Description 191
Table 295. MiscellaneousFiles.Filename 191
Table 296. Misc.Filename:Executable 192
Table 297. ModuleSurfaceArea.UserExtensions 193
Table 298. UserExtensions:UserId 193
Table 299. UserExtensions:Identifier 193
Table 300. DistributionPackage.Tools 195
Table 301. Tools.Header 196
Table 302. Header.Name 196
Table 303. Header.Copyright 196
Table 304. Header.License 196
Table 305. Header.Abstract 198
Table 306. Header.Description 198
Table 307. Tools.Filename 199
Table 308. Filename:OS 199
Table 309. Filename:Executable 199
Table 310. DistributionPackage.MiscellaneousFiles 201
Table 311. MiscellaneousFiles.Header 202
Table 312. Header.Name 202
Table 313. Header.Copyright 202
Table 314. Header.License 202
Table 315. Header.Abstract 203
Table 316. Header.Description 203
Table 317. MiscellaneousFiles.Filename 204
Table 318. Filename:Executable 204
Table 319. DistributionPackage.UserExtensions 206
Table 320. UserExtensions:UserId 206
Table 321. UserExtensions:Identifier 206

1 Introduction

This specification defines the overall architecture and external interfaces that are required for distribution of UEFI/PI source and binary files.

1.1 Overview

The packaging infrastructure defines the requirements for sharing code (source and/or binary) between firmware development teams using UEFI/PI elements.

This document describes the format of the distribution meta-data file, which has the following goals:

Distributing Package

An archive of "Packages".An archive of packages. Packages are collections of related include files, modules and libraries. Packages provide a method for defining industry standard objects used by modules, along with individual modules.

Distributing Libraries

Enable source distribution of libraries.

Distributing Modules

The final goal is to enable distribution of modules, both in source and binary form. Individual modules may be compiled and distributed in binary form to other users, which may be integrated easily into a platform's firmware image or option ROMs. Binary distribution is limited to FFS leaf section file types.

This document is intended for distribution package maintainers supporting UEFI/PI development. This content is required for the development of Distribution Package creation and installation tools and may also be useful for the development of custom Distribution Package tools that utilize the extension mechanisms defined in this specification. It may also be of interest if there is an issue with the creation, installation or removal of packages.

1.2 Related Information

The following publications and sources of information may be useful to you or are referred to by this specification:

- *Unified Extensible Firmware Interface Specification*, Version 2.1, Unified EFI, Inc, 2006, http://www.uefi.org.
- *UEFI Platform Initialization Specification*, Version 1.0, Unified EFI, Inc, 2006, http://www.uefi.org.
- *Extensible Markup Language (XML)* 1.0, Latest Edition, http://www.w3.org/TR/xml/
- *XML Schema Part 0*, Latest Edition, http://www.w3.org/TR/xmlschema-0/
- *XML Schema Part 1*, Latest Edition, http://www.w3.org/TR/xmlschema-1/

- *XML Schema Part 2*, Latest Edition, http://www.w3.org/TR/xmlschema-2/

1.3 Definition of Terms

The following terms are used throughout this specification.

BDS

PI Boot Device Selection phase.

BNF

BNF is an acronym for "Backus Naur Form." John Backus and Peter Naur introduced for the first time a formal notation to describe the syntax of a given language.

Distribution Content File

A file in ZIP format that holds all of the content (include files, libraries, modules) to be distributed.

Distribution Desription File

A file in XML format that describes the files in the content and the related information needed to install the file.

Distribution Package File

A file in ZIP format used for releasing and distributing source code or binaries. It contains a Distribution Content File and a Distribution Description File.

DXE

PI Driver Execution Environment phase.

DXE SAL

Special class of DXE module that produces SAL Runtime Services. DXE SAL modules differ from DXE Runtime modules in that the DXE Runtime modules support Virtual mode OS calls at OS runtime and DXE SAL modules support intermixing Virtual or Physical mode OS calls.

DXE SMM

Special class of DXE module that is loaded into the System Management Mode memory.

DXE Runtime

Special class of DXE module that provides Runtime Services

GUID and GuidValue

Globally Unique Identifier. A 128-bit value used to uniquely name entities. A unique GUID can be generated by an individual without the help of a centralized authority. This allows the generation of names that will never conflict, even among multiple, unrelated parties.

HII

Human Interface Infrastructure. This generally refers to the database that contains string, font, and IFR information along with other pieces that use one of the database components.

IFR

Internal Forms Representation. This is the binary encoding that is used for the representation of user interface pages.

Library Class

A library class defines the API or interface set for a library. The consumer of the library is coded to the library class definition. Library classes are defined via a library class .h file that is published by a package.

Library Instance

An implementation of one or more library classes.

Module Developer (MD)

A module developer is responsible for implementing new modules or maintaining existing modules. A module developer is responsible for describing all the information required to build a module and also describing the runtime behavior of a module. This description is called the Module Surface Area.

Module Surface Area

A full description of all of the elements a module produces and/or consumes that allow it to be integrated into a platform firmware image and executed as expected. These elements included binary file names, source file names, packages, libraries, protocols, PPIs, GUIDs, events, Platform Configuration Database elements, UEFI variables, UEFI System Configuration Tables, boot modes, HOBs, HII packages, and external variable and function names (including the module's entry point). Binary modules must minimally describe the dependencies required for a the module to execute correctly. Source modules must minimally describe their dependencies so that it can be compiled and linked into a binary module. Modules may also have runtime dependencies that must be satisfied in order for the module to execute correctly.

Module Type

All libraries and modules belong to one of the Module Types listed in Table 1, below.

Table 1. Module Type Names

Name	Description
BASE	Module that is environment neutral. Typically only used for libraries.
SEC	Module is the SEC and contains the reset vector.
PEIM	Module is a PEIM and depends on the PEI Services.
DXE_DRIVER	Module is a DXE driver and depends on the EFI Boot Services, EFI Runtime Services, and DXE Service Table.
DXE_RUNTIME_DRIVER	Module is a DXE driver and depends on the EFI Boot Services, EFI Runtime Services, and DXE Service Table. The module runs before and after **ExitBootServices** and produces **CreateEvent**, **EventGroupExitBootServices**, and **EventGroupVirtualAddressChange**. Code written to this module can run in physical or virtual mode.
DXE_SAL_DRIVER	Module is a DXE driver and depends on the EFI Boot Services, EFI Runtime Services, and DXE Service Table. The module runs before and after **ExitBootServices** and produces **CreateEvent**, **EventGroupExitBootServices**, and **EventGroupVirtualAddressChange**. Code written to this module can run in physical and virtual mode.
DXE_SMM_DRIVER	Module is a DXE driver and depends on the EFI Boot Services, EFI Runtime Services, and DXE Service Table. The module also runs in SMM mode and depends on the SMM Services Table.
UEFI_DRIVER	Module is a UEFI driver and depends on the UEFI Services.
UEFI_RUNTIME_DRIVER	Module is a UEFI driver and depends on the UEFI Services. The module runs before and after **ExitBootServices** and produces **CreateEvent**, **EventGroupExitBootServices**, and **EventGroupVirtualAddressChange**. Code written to this module can run in physical or virtual mode.
UEFI_APPLICATION	Module is a UEFI application and depends on the UEFI and Services.
DXE_CORE	Module is the DXE Core.
PEI_CORE	Module is the PEI Core.
USER_DEFINED	Module may be a binary image, in which the user will have an a priori knowledge of its usage.

Modules

All module references within this document refer to libraries, PI PEIMs, PI DXE drivers, UEFI drivers and UEFI applications. Libraries are mentioned separately, as they are not defined as a UEFI/PI image, but instead are linked to a UEFI/PI image during the build. Please note that a library can be module, there is no Module Type defined for LIBRARY.

Package

A package is a container. It can hold a collection of files for any given set of modules. Packages may contain Source Modules, containing all source files and descriptions of a module, or Binary Modules, containing PI FFS Sections and a description file specific to

linking and binary editing of features and attributes or both Binary and Source modules, Multiple modules can also be combined into a package. Multiple Packages can also be Bundled into a single Distribution Package.

Package Developer (PD)

The package developer is responsible for maintaining the contents of a package's declarations, which includes the package's name, description, copyright, licenses, and an optional set of library and module implementations. Packages may also contain elements that are available to the development of new modules by Module Developers. These include library class declarations, industry standard include files, include file extensions, GUID declarations, protocol declarations, PPI declarations, and platform configuration database entry declarations.

Package Distributor (PDist)

The Package Distributor is responsible for creating a distributable version of a package. This involves collecting a set of packages developed by Package Developers which includes modules developed by Module Developers and combine them into a Distribution Package.

Package Surface Area

A complete description of the elements a package provides to support building modules. This includes header files for industry standard specifications, library header files, include file paths within the package, GUID declarations, PPI declatations, Protocol declarations, and Platform Configuration Database element declarations. All of these package elements are available to Module Developers and can be used to describe the Module Surface Area of a module.

PCD

Platform Configuration Database.

PCD C Name

The symbolic name for a PCD Token that follows the ANSI C naming conventions for the name of a variable.

PCD Element

A single configurable element within the Platform Configuration Database, uniquely identified by a Token Space GUID and Token Number.

PCD Token Space GUID

The GUID value associated with a group of PCD Tokens. Using a GUID allows vendors to allocate their own Token Numbers for configuration elements that apply to their own modules, libraries or platforms without a centralized allocator. Within the Distribution Description file, a PCD Token Space GUID is referred to using the PCD Token Space GUID C Name.

PCD Token Space GUID C Name

A symbolic name for a PCD Token Space GUID value that follows the ANSI C naming conventions for the name of a variable.

PCD Token

A 32-bit unsigned integer that, along with the PCD Token Space GUID, uniquely identifies a PCD element. Within the Distribution Description File, a PCD Token is referred to using the PCD C Name.

PEI

Pre-EFI Initialization Phase.

Platform Configuration Database (PCD)

The collection of PCD elements that can be configured when building modules, libraries, or platform firmware images. These elements are identified by a Token Space GUID and Token Number. PCD elements are declared in packages by Package Developers. Modules Developers use PCD elements in the design of their modules to increase the portability of their modules to a wider array of platform targets. Platform Integrators set the values of PCD elements based on specific platform requirements. A Platform Integrator has many options when configuring PCDs for a specific platform. They may configure PCD elements to be set to static values at build time. They may also configure PCD elements so the binary image of a Module may be patched prior to integration into platform firmware images. They may also configure PCD elements so the binary image of platform firmware may be patched. They may also configure PCD elements so they can be accessed at runtime through the PCD services described in the PI 1.2 Specification.

Platform Integrator (PI)

The Platform Integrator is responsible for setting all the features required to build a FLASH image and boot a platform. This includes selecting pre-built modules, selecting modules implemented by Module Developers, selecting the library instances required to link the selected modules, setting the platform configuration database entry parameters for all the entries that are used by the selected modules and libraries, and selecting the FLASH layout required to store the pre-built module executables in a FLASH device.

PPI

A PEIM-to-PEIM Interface that is named by a GUID as defined by the PEI CIS.

Protocol

An API named by a GUID as defined by the UEFI specification.

Runtime Services

Interfaces that provide access to underlying platform-specific hardware that might be useful during OS runtime, such as time and date services. These services become active during the boot process but also persist after the OS loader terminates boot services.

SAL

System Abstraction Layer. A firmware interface specification used on Intel® Itanium® Processor-based systems.

SEC

Security Phase is the code that contains the processor reset vector and launches PEI. This phase is separate from PEI because some security schemes require ownership of the reset vector.

UEFI Application

An application that follows the UEFI specification. The only difference between a UEFI application and a UEFI driver is that an application is unloaded from memory when it exits regardless of return status, while a driver that returns a successful return status is not unloaded when its entry point exits.

UEFI Driver

A driver that follows the UEFI specification.

UEFI Specification Version 2.3

Current version of the UEFI specification released by the Unified EFI Forum.

Unified EFI Forum

A non-profit collaborative trade organization formed to promote and manage the UEFI standard. For more information, see www.uefi.org.

WORKSPACE

A project directory on a development system.

The following terms and descriptions are specific to the Module Surface Area defined within this specification.

Feature Flag

Feature flags are used within the source code as well as by a build system.

The feature flags represent how the module is coded. For each combination of feature flags supported by a module's source code, it is possible to produce a different binary surface area. PCD **`FEATURE_FLAG`** is a Boolean, so the relationship of each module surface area element to a binary can be expressed as a Boolean expression. The lack of a conditional expression implies TRUE, and the source code surface area element will be in all resulting binaries.

The Feature Flag can also be used by the build system to conditionally include or exclude different items (library classes, source files, etc.) from a build.

Consumes

Any item with a Usage Attribute of Consumes means that a module will not function without it. The module will either fail to execute, fail to build or fail to link if the item is not present when needed.

Produces

Any item with a Usage Attribute of Produces means that a module defines this specific instance of the item.

Sometimes_Consumes

Any item with a Usage Attribute of Sometimes_Consumes means that if the item is available, the item will be used, but the module may continue to function if the item is not available.

Sometimes_Produces

Any item with a Usage Attribute of Sometimes_Produces means that a module defines this specific instance of the item under certain conditions or under different execution paths.

ToStart

The protocol is required by the driver binding Start() function to make Start() succeed.

ByStart

The protocol is produced by the driver binding Start() function if it succeeds.

1.4 Target Audience

This document is intended for persons doing Package Development and Package Distributors.

1.5 Conventions Used in This Document

This document uses the typographic and illustrative conventions described below.

Note that each of the types with the prefix 'xs:' are built in to XML Schema. Full definitions of these simple types can be found online, http://www.w3.org/TR/xmlschema-0/#CreatDt.

1.5.1 Table Conventions

Table 2. Base XML Data Types

Name	Description
xs:boolean	A string containing ‘true’ or ‘false’
xs:decimal	A string containing numbers followed by a period followed by numbers Example: 1.23
xs:integer	A string that represents a signed 32-bit integer. The range is from -2147483648 to 2147483647
xs:nonNegativeInteger	A string that represents a positive 32-bit integer. The range is from 0 to 4294967295
xs:string	The set of finite-length sequences of characters that match Char
Char	#x9 | #xA | #xD | [#x20-#xD7FF] | [#xE000-#xFFFD] | [#x10000-#x10FFFF]
xs:normalizedString	The set of strings that do not contain the carriage return (#xD), line feed (#xA) or tab (#x9) characters.
xs:NCName	A string that start with a letter or an underscore and is followed by any combination of letters, digits, periods, hyphens, and underscores. No whitespace characters are allowed.
xs:anyURI	A finite-length character sequences which are legal URIs according to [RFC 2396], as amended by [RFC 2732]
xs:language	The language code entered must conform to RFC 1766.

Table 3. Value Types

Name	Description
xs:hexBinary	A string that contains any number of hexadecimal characters (0-9a-fA-F)* Example: F0040001
HexNumber	This simple type extends xs:hexBinary with a ‘0x’ prefix. Example: 0xF0040001
Md5Sum	A string of 32 hexadecimal characters that does NOT start with “0x.”
CFormatGuid	A straing containing a GUID in C data format, either new or simple style. Example Simple: {0xyyyyyyyy, 0xyyyy, 0xyyyy, 0xyy, 0xyy, 0xyy,0xyy, 0xyy, 0xyy, 0xyy, 0xyy} Example New: {0xyyyyyyyy, 0xyyyy, 0xyyyy, {0xyy, 0xyy, 0xyy,0xyy, 0xyy, 0xyy, 0xyy, 0xyy}} Where yy is a hexadecimal digit, whitespace characters are ignored.
RegistryFormatGuid	A string containing a GUID in registry format. Example: AF0DDA2E-EA83-480b-B2CE-FC0BB2F894C2
GuidValue	A string containing a GUID formatted as a RegistryFormatGuid.
GuidCName or CName	A C variable format word that can be used as a cross reference to

Table 4. Enumeration Types

Name	Description
Archs	A string that contains a single Arch type. The Arch type is restricted to the following set: **IA32** **X64** **IPF** **EBC** **Pattern = [A-Z]([a-zA-Z0-9])***
FamilyType	A string that contains a single Family type. The Family type is restricted to the following set: **MSFT** **GCC** **Pattern = [A-Z]([a-zA-Z0-9])***
GuidTypes	A string that contains a single GUID type. The GUID types are restricted to the following set: **Event** **File** **FV** **GUID** **HII** **HOB** **SystemTable** **TokenSpaceGuid** **Variable**

Name	Description
ModuleType	A string that contains a single module type. Module types are restricted to the following set: **BASE** **SEC** **PEI_CORE** **PEIM** **DXE_CORE** **DXE_DRIVER** **DXE_RUNTIME_DRIVER** **DXE_SAL_DRIVER** **DXE_SMM_DRIVER** **UEFI_DRIVER** **UEFI_RUNTIME_DRIVER** **UEFI_APPLICATION** **USER_DEFINED**
PcdDatumType	A string that contains a single PCD Data Type. PCD Data Types are restricted to the following set: **UINT8** **UINT16** **UINT32** **UINT64** **VOID*** **BOOLEAN**
PcdItemTypes	A string that contains one PCD Entry item types. The PCD Entry item types are restricted to the following set: **FeaturePcd** **FixedPcd** **PatchPcd** **Pcd** **PcdEx**

Name	Description
SupportedOs	A string that contains one Supported Operating System type. The Supported Operating System types are restricted to the following set: **Win32** **Win64** **RedHat32** **RedHat64** **SuSE32** **SuSE64** **Generic** **GenericWin** **GenericNix** **Pattern = [A-Z]([a-zA-Z0-9])***

Table 5. List Types

Name	Description
ArchListType	A string that contains one or more CPU architectures separated by spaces. The CPU architectures are restricted to the following set: **IA32** **X64** **IPF** **EBC**
GuidListType	A string that contains one or more GUID types separated by spaces. The GUID types are restricted to the following set: **Event** **File** **FV** **GUID** **HII** **HOB** **SystemTable** **TokenSpaceGuid** **Variable**
ModuleListType	A string that contains one or more module types separated by spaces. The module types are restricted to the following set: **BASE** **SEC** **PEI_CORE** **PEIM** **DXE_CORE** **DXE_DRIVER** **DXE_RUNTIME_DRIVER** **DXE_SAL_DRIVER** **DXE_SMM_DRIVER** **UEFI_DRIVER** **UEFI_RUNTIME_DRIVER** **UEFI_APPLICATION** **USER_DEFINED**

Name	Description
PcdItemListType	A string that contains one or more PCD Entry item types separated by spaces. The PCD Entry item types are restricted to the following set: **FeaturePcd** **FixedPcd** **PatchPcd** **Pcd** **PcdEx**

Table 6. Table Description

Description	This is the description of this element
Required	YES | NO – Is this element required by the parent element
Data Type	[Attribute | Element] – [Complex | Data Type]
Data Constraints	What are the limitations or possible values for this element
Examples	Either N/A or a hypothetical example

1.5.2 Pseudo-Code Conventions

Pseudo code is presented to describe algorithms in a more concise form. None of the algorithms in this document are intended to be compiled directly. The code is presented at a level corresponding to the surrounding text.

In describing variables, a list is an unordered collection of homogeneous objects. A queue is an ordered list of homogeneous objects. Unless otherwise noted, the ordering is assumed to be First In First Out (FIFO).

Pseudo code is presented in a C-like format, using C conventions where appropriate. The coding style, particularly the indentation style, is used for readability and does not necessarily comply with an implementation of the Extensible Firmware Interface Specification.

1.5.3 Typographic Conventions

This document uses the typographic and illustrative conventions described below:

This document uses the typographic and illustrative conventions described below:

Plain text	The normal text typeface is used for the vast majority of the descriptive text in a specification.
Plain text (blue)	In the online help version of this specification, any plain text that is underlined and in blue indicates an active link to the cross-reference. Click on the word to follow the hyperlink. Note that these links are *not* active in the PDF of the specification.
Bold	In text, a **Bold** typeface identifies a processor register name. In other instances, a **Bold** typeface can be used as a running head within a paragraph.

Italic	In text, an *Italic* typeface can be used as emphasis to introduce a new term or to indicate a manual or specification name.
BOLD Monospace	Computer code, example code segments, and all prototype code segments use a **BOLD Monospace** typeface with a dark red color. These code listings normally appear in one or more separate paragraphs, though words or segments can also be embedded in a normal text paragraph.
Bold Monospace	In the online help version of this specification, words in a **Bold Monospace** typeface that is underlined and in blue indicate an active hyperlink to the code definition for that function or type definition. Click on the word to follow the hyperlink. Note that these links are *not* active in the PDF of the specification. Also, these inactive links in the PDF may instead have a **Bold Monospace** appearance that is underlined but in dark red. Again, these links are not active in the PDF of the specification.
Italic Monospace	In code or in text, words in *Italic Monospace* indicate placeholder names for variable information that must be supplied (i.e., arguments).
Plain Monospace	In code, words in a Plain Monospace typeface that is a dark red color but is not bold or italicized indicate pseudo code or example code. These code segments typically occur in one or more separate paragraphs.

See the references sections in the *UEFI 2.1 Specification* for a complete list of the additional documents and specifications that are required or suggested for interpreting the information presented in this document:

The *UEFI Specification* is available from the UEFI web site http://www.uefi.org/specs.

See the references sections in the UEFI Platform Initialization Specification Version 1.0 for a complete list of the additional documents and specifications that are required or suggested for interpreting the information presented in this document:

The *UEFI Platform Initialization Specification* is available from the UEFI web site http://www.uefi.org/specs.

2
Design Discussion

UEFI and PI components made of different collections of related objects are called packages. Individual packages may be updated without having to update other packages, so as tools, or architecture specifications are updated, only the relevant packages need to be updated. If an architecture specification changes, the corresponding distribution package containing the architecture definition package can be downloaded, without downloading an entire development tree. If new tools are released, a new tools distribution package can be distributed without disturbing the contents of the other packages. As industry standards evolve, updates to the standards (typically defined in header files) can be released, again, without having to get an entirely new tree.

This document uses two different concepts for the term "package." The first is a distribution package; for clarity, this document will use the term "distribution" when referring to a distribution package. The other usage of package is a collection of related objects – Includes, Libraries and Modules. A distribution package may contain zero or more of the packages, while each package may contains zero or more modules. (Why zero? – A package can consist of nothing but industry standard headers. While most industry standard headers may need to provide library routines, there is no requirement for providing them.) Each package typically has it's own folder or directory.

2.1 Purpose

In order to provide a consistent process for sharing drivers and code, this specification introduces the concept of a Package Surface Area and a Module Surface Area which document the surface area of packages and modules. These surface areas include many concepts from the UEFI Specification and and PI Specification as well as a Platform Configuration Database.

The Platform Configuration Database is the collection of elements that can be configured when building modules, libraries, or platform firmware images. These elements are identified by a Token Space GUID and Token Number. PCD elements are declared in packages by Package Developers. Modules Developers use PCD elements in the design of their modules to increase the portability of their modules to a wider array of platform targets. Platform Integrators set the values of PCD elements based on specific platform requirements. A Platform Integrator has many options when configuring PCDs for a specific platform. They may configure PCD elements to be set to static values at build time. They may also configure PCD elements so the binary image of a Module may be patched prior to integration into platform firmware images. They may also configure PCD elements so the binary image of platform firmware may be patched. They may also configure PCD elements so they can be accessed at runtime through the PCD services described in the PI Specification.

A PCD element is identified by a Token Space GUID and a Token Number. The Token Space GUID is a GUID value that is associated with a group of PCD Tokens. Using a GUID value allows vendors to allocate their own Token Numbers for configuration elements that apply to their own modules, libraries or platforms without a centralized allocator. Token Space GUID values must be declared in a surface area for a package along with a symbolic name for the GUID. This symbolic name is called the Token Space GUID C Name. Token Number must also be declared in the surface

area for a package along with the symbolic name for the PCD Token. This symbolic name is called the PCD C Name. If a module uses a PCD element, then that usage is described in the surface area of the module, and the PCD element in the surface area of a module is identified using the symbolic name for the Token Space GUID and the symbolic name for the PCD Token. This means the module surface area only references the Token Space GUID C Name and the PCD C Name. This is much easier to read and maintain than the GUID values and Token Numbers.

The surface area of a package is the complete description of the elements a package provides to support building modules. These elements include:

- Header files for industry standard specifications
- Library header files
- Include file paths within the package
- GUID declarations
- PPI declarations
- Protocol declarations
- Platform Configuration Database element declarations

The surface area of a module is a full description of all of the elements a module produces and/or consumes that allow it to be integrated into a platform firmware image and executed as expected. These elements included:

- Binary file names
- Source file names
- Packages
- Libraries
- Protocols
- PPIs
- GUIDs
- Events
- Platform Configuration Database elements
- UEFI variables
- UEFI System Configuration Tables
- Boot modes
- HOBs
- HII packages
- External variable and function names, including the module's entry point

Binary modules must minimally describe the dependencies required for a the module to execute correctly. Source modules must minimally describe their dependencies so that it can be compiled and linked into a binary module. Modules may also have runtime dependencies that must be satisfied in order for the module to execute correctly.

The surface area of a module describes how a module can be integrated quickly into a new project. The surface area descriptions can be broken down into source, binary, and library integration.

Key module surface area features include:

- The module surface area describes all things visible to the system from a binary module.
- The module surface area describes information needed to construct and link modules.
- The module surface area does not describe everything in the source code of a module.
  - This is very complicated and C source code does a good job of this.
- A module must inherit the module surface area of the library linked to that module.
- The module surface area describes only the library class and not the library instance.
  - The library instance linked to a module or library is determined at link time.
  - The complete module surface area of a module can only be determined after library instances have been selected.

The challenge with integrating at a source level is that the surface area of the module might contain conditional statements that define different binaries based on the build options that are chosen. A binary needs to describe how it was built; it is not necessary to describe how it could be built if no source were included. If a binary supports patching, the binary also needs to describe the option and how to patch it, while the source surface area would need to describe only the option.

While it's possible to categorize a library as either a source or binary integration point, subtle differences warrant libraries having their own category. The primary difference is that libraries are often included into modules. This means the consumer of a library must inherit the module surface area of the libraries linked to the module.

One of the biggest challenges with the module surface area will be making the initial description correct and maintaining this correctness as the module is maintained. For this reason, the build environment should be strongly tied to the surface area description. For example, if you don't have a HOB element in the module's surface area description, using HOBs would prevent the module from compiling until the surface area description was updated. A large number of the surface area grammar elements are GUIDs, so tying compile time GUID resolution to a module surface area would go a long way toward making sure the surface area is always defined correctly. It's not practical to make a 1:1 mapping between the surface area grammar and making the module compile, but having a 60%-80% mapping will give the developer the mind set of updating the surface area description when changes are made to the driver.

2.2 Surface Area Description

The surface area consists of the following parts:

Usage Description

Describes how the element is used.

Element

Describes the surface area element.

Conditional

Optional definitions of what PCD feature flags this element is valid for.

Default

An optional, recommended default value for the element. This is generally used by the platform integrator as a guide in selecting what value should be used on a given platform for a given PCD value.

Help Text

Each surface area element can optionally include help text.

The usage description defines how an element is coded or used by a module. The usage includes production or consumption and whether the element is always or only sometimes used by the driver. The element describes exactly what the module is producing or consuming.

The optional conditional statement can be used to qualify the surface area statement to be valid only if the module is built a certain way. If no conditional exists, it is assumed to be TRUE, and the surface area statement is valid for all ways the module can be built. The conditional statement is a Boolean expression of PCD Feature Flags.

2.2.1 Conditional Surface Area Entries

Some surface area entries might be in the source code, but not in the resulting binary. The module surface area of the source code describes what's in the source code, while the module surface area of a binary describes what and how the module was built. If a module does different things for different processor types, the surface areas of the resulting binaries are different. Thus, it's important to think of the module surface area in terms of the source code and the binary. The surface area description for the source code is easy — if an element is in any of the source code, it must be in the module surface area for the source code. The module surface area for a binary is more complex. The build system can construct code for different processor architectures in radically different ways, if needed, and the resulting module surface area for the binary needs to reflect only what was constructed.

It's also possible that a PCD Feature Flag could cause surface area elements that are in the source code to not be in the resulting binary.

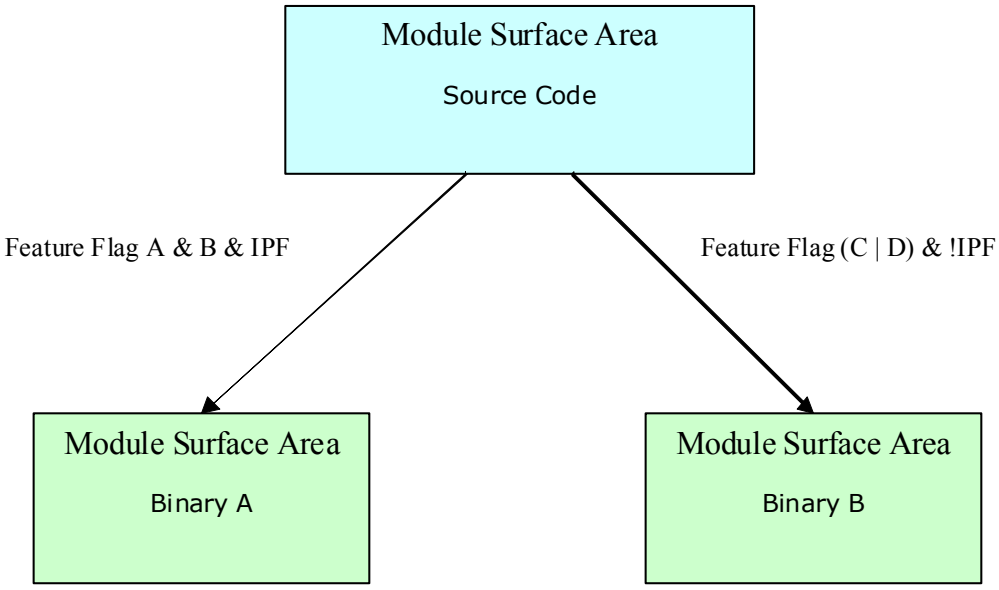

Figure 1. Module Surface Area Differs for Different Binaries

Figure 1 illustrates how the surface area is different for the source code and the different binaries that can be produced from common source code. In the above example, the surface area of the binary differs based on processor type: for Itanium® Processor binaries you get surface area A, and for non-Itanium® Processor binaries you get surface area B. A and B are both subsets of the source code surface area, as only things that are in the source code can end up in the binary.

The feature flags must represent how the module is coded. For each combination of feature flags supported by a module's source code, it is possible to produce a different binary surface area.

PCD "FeaturePcd" is a Boolean, so the relationship of each module surface area element to a binary can be expressed as a Boolean expression. The lack of a conditional expression implies TRUE, and the source code surface area element will be in all resulting binaries.

2.2.2 Surface Area Inheritance

The module surface area for a driver describes how the driver is coded. The previous section describes how PCD "FeaturePcd" can impact the resulting surface area of the binary produced when the module is compiled and linked. The surface area for the binary must also inherit surface area elements from any linked library instance.

The surface area for a driver specifies only the class of the library.

2.2.3 Source Code Surface Area

The distinguishing feature of the source code surface area is that it outlines a set of options in the form of PCD-based build flags. The runtime surface area is a proper subset of the build-time surface area.

2.2.4 Informational Surface Area

Example: PCD:UsbEnable flag. There may be a driver that does not use the UsbEnable flag as this driver is an UEFI driver that produces USB interfaces. If the platform has no USB, this driver should not be included in the build. There may be other drivers, such as ICH USB, that need to conditionally compile to not program certain registers based on UsbEnable. The ICH driver would define the PCD flag in the normal location. The UEFI USB driver would describe an informational surface that was a parallel of the build-time surface area to help people whether it should be included in the build.

2.2.5 Conditional Descriptions

PCD entries of the type build can also be used to describe the surface area in a conditional way. All PCD values must be available to all build tools, usable in all build and packaging files. This is in addition to the requirement to include PCD entries as static data, dynamic data, or binary patchable data in a module.

2.2.6 Binary Surface Area

The binary surface area defines the surface area of a module when it executes its code. The binary surface area is a subset of the source code surface area. Descriptions of possible build options are not needed, and only elements consistent with the current compile options are defined.

2.2.7 Hardware Surface Area

This area is not fully explored in this version of the specification. Areas of interest include:

- "Defining hardware settings for CMOS (and other indexed I/O devices), I/O, memory mapped, and PCI hardware devices.
- "Usage to define which base registers a driver assumes are already programmed in a chip that is being programmed.
- "May be used to define which PCI devices a UEFI or DXE driver could support.

2.3 Surface Area Grammar

The surface area grammar defines how a module either produces or consumes public interfaces or elements. There are seven universal usage descriptions (see Usage Description table below) that can be applied to any of the grammar elements. Usage descriptions define how a module either produces or consumes a public interface or element. A module may require (**`CONSUMES`**), conditionally require (**`SOMETIMES_CONSUMES`**), or need an interface to bind the driver using the UEFI driver model (**`TO_START`**). A module may always produce (**`PRODUCES`**), conditionally produce

(**SOMETIMES_PRODUCES**), or produce an interface following the UEFI driver model (**BY_START**).

Table 7. Usage Description

Name	Description
CONSUMES	Element is required for the driver to function. This means the element is part of the dependency expression of a module if the element represents a PPI or Protocol. For the most part, any GUID, Protocol, PPI or PCD that doesn't install (PRODUCES) something will be one form of CONSUMES.
SOMETIMES_CONSUMES	Element is consumed by the driver only if the element exists.
PRODUCES	Element is always produced (installed) by the driver
SOMETIMES_PRODUCES	Element is conditionally produced by the driver
TO_START	Protocol is required by driver binding **Start()** function to make **Start()** succeed.
BY_START	Protocol is produced by driver binding **Start()** function if it succeeds.

PEI and DXE drivers define their dependencies in the terms of the PPIs or protocols that they require to be present in the system before the given module would be executed. While this works well for ordering the execution order of modules it is not a sufficient description to easily port a module into a new project. The grammar defined in the table below, can be used to define which public interfaces a module produces or consumes. Each element of the grammar can be clarified by combining it with a usage description.

For each element, Table 8 gives the type of name required to specify the instance of the element and a description of the element.

Table 8. Surface Area Grammar Summary

Element	Name Type	Description
Package	GUID	Packages are the basic unit of source code and binary module distribution. A single package can imply (depend on) other packages. Only the leaf package needs to be specified in this case. Packages are named by GUID (and version.) Packages generally make include files and libraries available to modules.
LibraryClass	Keyword	Defines a class of libraries that provide similar features (prototypes are defined in a library class header file provided by a package.) There may be more than one library instance that can provide the feature. (See Libraries.)
Libraries	GUID	A module needs to explicitly state the library functions it uses (by including the library class header file that contains the function prototypes.) A Keyword specifies the class of the library; while a GUID specifies the name of the library instance. Multiple libraries can support the same class, so it's not possible to declare the library instance in the source code. The build environment must know the mapping of library classes to actual library instances. The library instances are selected by the Platform Integrator. Library Instances always provide (**PRODUCES**) library functions, they may required (**CONSUMES** or **SOMETIMES_CONSUMES**) other library functions in order to function.
Protocol	GUID	Specifies whether a protocol named by GUID is **CONSUMES**, **SOMETIMES_CONSUMES**, **PRODUCES**, or **SOMETIMES_PRODUCES**.
ProtocolNotify	GUID	Specifies whether a module requires or consumes a protocol, named by a GUID, via a register protocol notify mechanism. Valid usage can only be **SOMETIMES_CONSUMES**.
Ppi	GUID	Specifies whether a PPI named by GUID is **CONSUMES**, **SOMETIMES_CONSUMES**, **PRODUCES**, or **SOMETIMES_PRODUCES**.
PpiNotify	GUID	Specifies whether a module requires or consumes a PPI, named by GUID, via a register protocol notify mechanism. . Valid usage can only be **SOMETIMES_CONSUMES**.
Event (Create)	GUID	Module has an event that is waiting to be signaled (**CONSUMES** or **SOMETIMES_CONSUMES**.) The event is named by GUID. UEFI 2.0 added **CreateEventEx** and maps the existing EFI events back to GUIDs. **EVENT_GROUP_GUID** **EVENT_TYPE_PERIODIC_TIMER** **EVENT_TYPE_RELATIVE_TIMER**

Element	Name Type	Description
Event (Signal)	GUID	Module will signal all events in an event group (**PRODUCES** or **SOMETIMES_PRODUCES**.) Event groups are named by GUID just like **CreateEvent**.
FileName	GUID	FV Filename GUID.
GUID	GUID	A GUID that is not described by one of the other types in this table.
Pcd.$(database)	PCD Item:{Type}	Platform Configuration Database (PCD) represents a set of GUIDed databases containing sets of items. Packages publish PCD basenames. A module can be built with a hard-coded value for an Item, with the Item being dynamic, or with the Item being a value that can be binary patched. $(database) is a GUID. No GUID implies the default PCD database.
	{Type}	Description
	FeaturePcd	The PCD Item represents a feature flag for the module. Features can be selected only at build time. Items of type **FeaturePcd** are used to conditionally construct a module surface area that is produced as a result of the build. A **FeaturePcd** must be a Boolean.
	FixedPcd	PCD Item is only a build-time option and can not be Dynamic or binary patched into the module (**PatchPcd**).
	PatchPcd	Module Surface Area for Source Code: PCD Item {Type} is set to a default value at build time and the binary of the module can be patched to update the value. **FixedPcd** implies the build assigns a value when the module is compiled. **PatchPcd** implies the build patches the module binary as part of the build option. The **PatchPcd** is most useful if you don't have source code to a module that you are including in the build.
		Module Surface Area for Binary: The binary must also describe the location within the binary to patch. The surface area must describe the file offset into the uncompressed PE32+ image to associate with the PCD entry. This can be a UINT32 (max PE32+ image size) offset. It may also be useful to specify the label name in the image of the PCD entry.
	Pcd	Module Surface Area for Source Code: This dynamic form implies the source code contains a macro that will be resolved via a build option into **FixedPcd**, **PatchPcd**, or **Pcd**. If no {Type} is present, it defaults to **Pcd**. This allows the user of the module (Platform Integrator) to pick the type as the module is coded to allow all three.

Element	Name Type	Description
		Module Surface Area for Binary: PCD Item {Type} is found via a PCD PPI in PEI or PCD protocol. The token that matches the PCD entry is either generated by the build (only unique to the build) or is from the default database.
	PcdEx	PCD Item {Type} is found via PCD PPI in PEI or PCD protocol in DXE. Any PCD token database is supported.
Variable	GUID:String{offset}	An EFI variable described by a GUID string pair. No {offset} value implies the entire variable is used.
	Byte	The first {offset} entry represents the byte offset to the start of the data. The names Byte is not part of the syntax and actual numbers must be used.
SystemTable	GUID	An EFI system table entry identified by a GUIDed system table is **CONSUMES**, **SOMETIMES_CONSUMES**, **PRODUCES**, or **SOMETIMES_PRODUCES** by a module.
ModuleType	ModuleType	Type of module being developed. An instance of a library can support more than one ModuleType.
BootMode	BootMode	A module uses BootMode to define the boot modes it can set (**PRODUCES** or **SOMETIMES_PRODUCES**) or the boot modes it supports (**SOMETIMES** or **SOMETIMES_CONSUMES**.) Not having this element implies no boot mode is set and all boot modes are supported.
Hob	HobType	Represents a HOB that is being produced or consumed. A HobType of **GUID_EXTENSION** requires that the name of the GUID being used is also required.
FormSet	GUID	IFR form set was added (**PRODUCES** or **SOMETIMES_PRODUCES**) by this module via the HII protocol.
Extern	C Identifier Name	Libraries can depend on externally defined symbols.

2.4 Distribution

A Distribution Package File is a file (ZIP format) that contains two files: a Distribution Description File (XML format) and a Distribution Content File (ZIP format). Installations and updates using these Distribution Package Files follow rules to prevent package collisions and permit relocating packages (and modules) to different directory names during an installation.

The Distribution Description File provides a description of the content provided by the Distribution Content File. The content described includes every package, module and tool, as well as any miscellaneous items. Additionally, the package and module surface area descriptions within the Distribution Package Desription File document any dependencies and describe how a module is coded, what resources the module produces or consumes, and, in the case of binary modules, how the binaries in the module were built.

2.5 Tracking

Packages and all modules within any given package are identified using GUID values and VERSION numbers. The surface area descriptions use these GUID and VERSION values to describe dependencies. If a module depends on a specific package, the package's GUID version should be specified in the module's package dependencies section to identify the package. If the module is coded so that it will not depend on a specific version of a package, the VERSION can be omitted. However, if a module requires a specific version of a package to satisfy a dependency, then both the GUID and VERSION of the package must be listed. If VERSION is omitted and there are more than one packages with the same GUID installed in the workspace the package with the highest VERSION value will be used.

Additionally, this document introduces the concept of modules that are coded against classes of libraries, rather than individual library instances. This method permits the platform integrator to select specific instances of libraries that satisfy the library classes a module was coded against.

3
Distribution Package File

The Distribution Package File format is the format of the file used in distributions. Instead of building or specifying an entirely new archive format, the Distribution Package File simply extends a widely accepted industry standard ZIP file format. The ZIP format already supports most of the features or capabilities required by Distribution Package Files, including:

- Compression
- Cross-platform support
- Encryption

The Distribution Description File is a meta-data file that all distributions are required to contain. The Distribution Description File is an XML formatted file with a .pkg file extension that resides in the root directory of the Distribution Package File.

The Distribution Package File also holds all of the Include files, Libraries and or Modules. These files are contained in the Distribution Content File, a ZIP formatted file with a .content file extention that resides in the root directory of the Distribution Package File. Keeping the meta-data information about what is being distributed and the content separate provides a clean separation between the two distinct elements.

3.1 Distribution Requirements

The Distribution Package File containing the Distribution Description File and the Distribution Content File must have the file extension of **`.dist`**.

A Distribution Package File may contain:

1. Zero or more packages
2. Zero or more modules.
3. Zero or more tools.
4. Additional files that do not match either a package surface area declaration, module surface area or tools.

The Distribution Package File consists of two parts:

- The Distribution Description File, which is the XML formatted data describing the Distribution Content.
- The Distribution Content File, which is the the compressed package tree.

Any additional files within the Distribution Content File may be ignored.

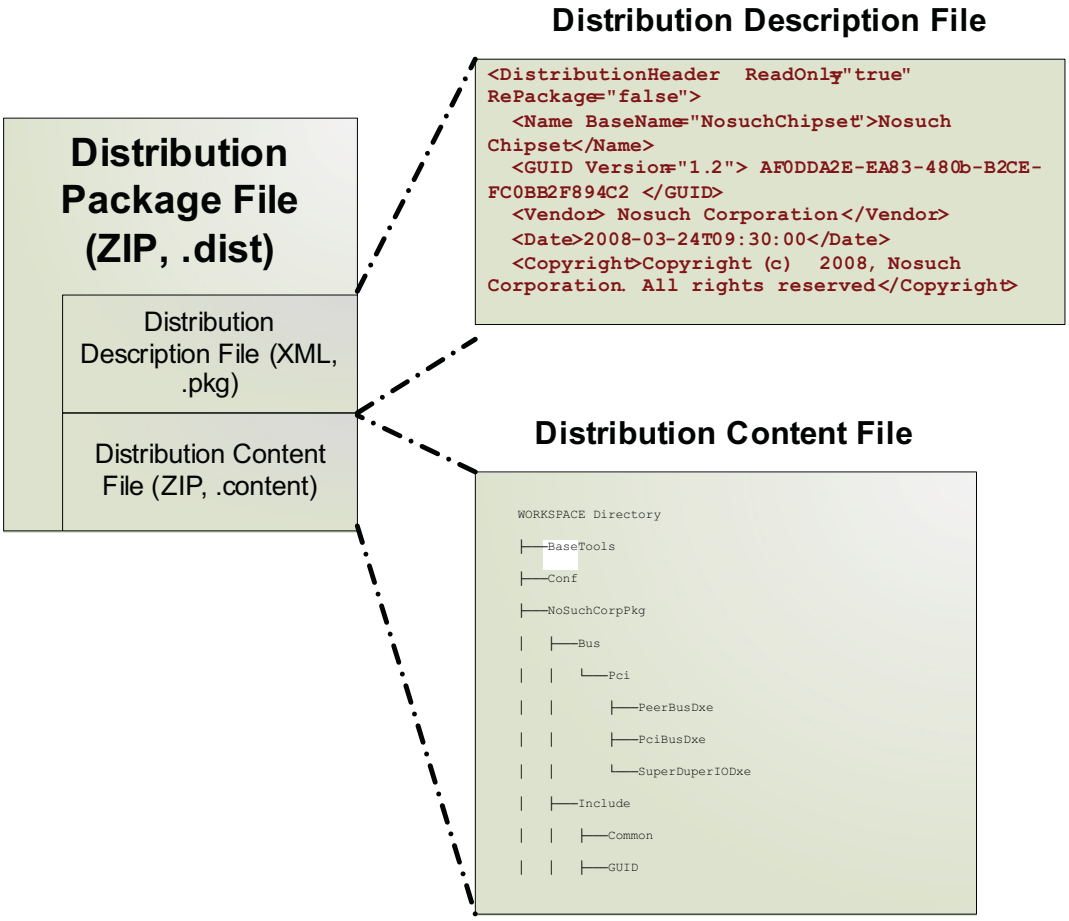

Figure 2. Distribution Package File layout

3.1.1 Creating a Distribution Package File

Normally, a Distribution Package File is created from a directory tree located in a directory on a developer's workstation. The typical directory tree is a hierarchy of directories and files that are a collection of declarations and modules. This directory tree may contain an Includes directory, a Library directory which has library modules (instances) directories beneath it, and module directories that contain drivers, applications and/or other non-library modules. Figure 3, below, shows an example directory tree for the NoSuchCorpPkg.

```
WORKSPACE Directory
+---BaseTools
+---Conf
+---NoSuchCorpPkg
¦   +---Bus
¦   ¦   +---Pci
¦   ¦       +---PeerBusDxe
¦   ¦       +---PciBusDxe
¦   ¦       +---SuperDuperIODxe
¦   +---Include
¦   ¦   +---Common
¦   ¦   +---GUID
¦   ¦   +---Library
¦   ¦   +---Protocol
¦   +---Library
¦   ¦   +---NoSuchHookStatusCodeLib
¦   ¦   +---SmmRuntimeDxeReportStatusCodeLib
¦   +---Universal
¦       +---DualSegmentCfgPei
¦       +---DualSegmentCfgDxe
¦       +---StatusCode
¦           +---Dxe
¦           +---Pei
+---MdePkg
+---MdeModulePkg
```

Figure 3. NoSuchCorpPkg Directory Tree

A Distribution Package File may be created by the following steps:

1. Determine the set of files that will be distributed, including packages, modules, tools, and miscellaneous content.
2. Create the Distribution Description File, and complete the Distribution Header Information.
3. For each package that will be included in the Distribution Content File,
   - If a supported "declaration" file, exists for the package, it can be used as a starting point - otherwise, the information will need to be filled in manually.
   - Specify any Library Classes, GUIDS, Protocols or PPIs that are declared by header files in the package.
   - List the reference names of the modules that are to be included in this package - every reference name provided in this list must have a corresponding module surface area section in this file.
   - Specify any PCDs that are declared within the package. Typically, these are defined in a specification that uniquely identifies the declared PCDs, i.e., a NoSuch Package Specification would define the PcdTokenSpaceGuidCName: gEfiNoSuchPkgTokenSpaceGuid (the GUID is for this is declared in step b, above) and individual PCD information for each PCD in the spec. Each PCD must be declared in every PCD section type that it might be coded for, i.e., PcdsFixedAtBuild, PcdsPatchableInModule and PcdsDynamic.
   - Specify the location of any Include folders that are provided by this package.
   - Include all top level header files that are to be included in the Include folders. For example: "Include/Ppi/CpuIo.h"

- Add any other package files that are to be included and not listed in e. Module information is handled separately, so do not include them here. Listing the supported declaration file is not required. Test "platform" files, used for testing a build of modules that are part of a package may be included for reference.
- Add the package information to the Distribution Description File.

4. For each module that will be included in the Distribution Content File,
   - If a supported module description file exists for the module, it can be used as a starting point, otherwise, the information will need to be entered manually.
   - Complete the Module header information.
   - Include all the files referred to by the module description file. Add filenames to either the SourceFiles or BinaryFiles section as appropriate, using file paths that are relative to the root of the module. For example: "Ia32/CpuPause.asm"
   - Include all files that are part of a module, but not listed in the module description file. Listing the module description file is not required.
   - Add the module information to the Distribution Description File.
5. For all tools that are going to be distributed,
   - Complete the Tool section header information
6. For any thing not listed in steps 2, 3 or 4,
   - Complete the MiscellaneousFiles section
   - Complete a File entry for each file that will be included in the distribution not already entered.
   - Complete any additional information for the MiscellaneousFiles section using freeform text or XML for the content.
7. Create the Distribution Package File.
   - Generate the Distribution Content File from the content.
   - Compute the MD5 sum of the Distribution Content and update the Signature field in the Distribution Description File's header.
   - Add the Distribution Description Fle and Distribution Content File to the Distribution Package File.

3.1.2 Installing a Distribution Package File

This section covers the general steps need to install a Distribution Package File on a developer's workstation. Tools may be required to have an a priori knowledge of the developer's development project space.

1. The user may need to select a build meta data format for generating build specific meta-data.
2. The tool need to generate a 'database' of content installed in the project space.
3. For each package declaration section
   - Test the dependencies for the declaration section - See Distribution Rules and Definitions, below.
   - Specify an alternate installation location within the project space, if needed.
   - Create the build specific package declaration files, if required.

- Extract any files within this section to the specified location.
- Update any distribution management "database" to point to a new location - only required if the default path was not used to extract the file(s.)

4. For each module information section,
   - Test the dependencies for the module section - See Distribution Rules and Definitions, below.
   - Specify an alternate installation location within the project space, if needed.
   - Create the build specific module description files, if required.
   - Extract any files within this section to the specified location.
   - Update any distribution management "database" to point to a new location - only required if the default path was not used to extract the file(s.)
5. For a Tool section,
   - Specify an alternate installation location, if needed.
   - Extract any files within this section to the specified location.
   - Update any distribution management "database" to point to a new location - only required if the default path was not used to extract the file(s.)
6. For a MiscellaneousFiles section,
   - Specify an alternate installation location, if needed.
   - Extract any files within this section to the specified location.
   - Update any distribution management "database" to point to a new location - only required if the default path was not used to extract the file(s.)
7. Store the Distribution Description File in a directory as specified by any tools.
   - If an alternate name has been specified, store the file using the alternate name and .pkg extension in a directory as specified by any tools.
   - If no alternate name was provided, and if a Distribution Description File already exists, then rename the Distribution Description File filename with _<Distribution Package GUID> appended prior to the .pkg extension. If the name already exist, ask the user for an alternate name, and use that name instead.

3.1.3 Removing a Distribution

This function will remove the selected distribution package from the project directory.

1. From distribution management "database", select the Distribution Description File that matches the Distribution Package File that will be removed.
2. Test the Package against dependencies specified in all of the supported meta-data infrastructure files in the project.
   - If any build meta-data file requires this package, let the user know that meta-data files require this package, and give them an option to abort the removal. If they choose to continue to remove the package, print the list of meta-data files that depend on the package, and inform the user that a list of dependent meta-data files that are no longer valid is in a file.
3. Remove all files specified in the Distribution Description File using the extracted information if the files were stored in an alternate location.

3.2 Distribution Rules and Definitions

Definitions and rules for creating, installing, updating, and removing distributions within a project WORKSPACE. The terms GUID and PackageGuid in the rules below refer to the value of a GUID. The terms Version and PackageVersion in the rules below refer to a version value.

Table 9. Distribution Rules & Definitions KEY

Letter	Example Definitions
W	A workspace
M	A module
D	A Distribution
P	A package
L	A location
G	Other Distribution
Q	Other Package

1. A module **M** is said to depend upon a package **P**, if and only if there exists a tuple (PackageGuid, PackageVersion) in the set **M->PackageDependencies** for which **P->GUID == PackageGuid**, and if PackageVersion is not empty, then **P->Version == PackageVersion**.
2. A distribution **D** is said to depend on a Distribution **G**, if and only if there is a module in a package in **D** that depends on a package in **G**.
3. A distribution **D** is said to depend on a package **P**, if and only if there is a module **M** contained in **D** that depends on **P**.
4. A distribution **D** may be installed into the WORKSPACE **W**, if and only if for each module **M** in **D**, **M**'s dependencies are met by the packages in **W**.
   - If the dependencies are not met, then no part of distribution **D** will be installed. It is not legal to partially install a distribution into the WORKSPACE.
5. A distribution **D** may be removed from the WORKSPACE **W**, if and only if for each module M in WORKSPACE **W**, and for each package **P** in **D**, **M** does not depend on **P**.
   - If there is some dependency on **D**, then no part of **D** may be uninstalled from **W**. It is not legal to partially uninstall a distribution from the WORKSPACE.
6. When installing a distribution **D** into WORKSPACE **W**, for each package **P** in **D**, allow the user to install in **P**'s default location, or choose a new location **L** (which must be unoccupied) within the WORKSPACE. Record this location **L** in the database. Each package **P** in **D** will be recorded in the database, associated with the GUID of **D**, as well as the actual install location **L**. (So we will know which distribution each package belongs to.)
7. When installing a distribution **D** into WORKSPACE **W**, if there exists a package **P** in **W**, and **P** is in **D**, then the user must be prompted to choose a location that does not collide with the location of P in WORKSPACE **W**. We will end up with two instances of **P** in **W** at two distinct locations.
8. A distribution **D** may replace a distribution **G** in the WORKSPACE **W**, if and only if for each module **M** contained in **W**, if **M** depends on a package **P**, and **P** is only contained in **G**, then there must exist a package **Q** in **D**, such that the dependency of **M** can be satisfied by **Q**.

- The net effect is that **G** is removed and **D** is installed, in one operation. The normal rules for installing **D** still apply - the dependencies of the modules of **D** must be satisfied. After the replacement, it must be the case that all the modules' dependencies in the WORKSPACE are satisfied.
- If we find that the replace operation is not permitted, then the user may install **D** and keep **G**. Next, the user may "port" to **D** every package **P** in **G** on which a module **M** in **W** depends. Once all the dependencies can be resolved without the packages in **G** the user can remove **G**.

9. A special case of the above rule is that a distribution **D** may be reinstalled into the WORKSPACE. (This would allow the user to get a fresh copy, or change the location in the WORKSPACE where one or more of the packages of **D** are installed.)
10. When a distribution **D** is removed from the WORKSPACE **W**, for each package **P** in **D**, we will remove **P** from **W**.
11. If a package **P** belongs to a distribution **D**, then it is not legal to remove P from the WORKSPACE **W** unless **D** is removed from **W**.
12. A package **P** may be removed from the WORKSPACE, provided there does not exist a distribution **D** that contains **P**. (Newly created or cloned packages will not exist within a distribution, and thus may be removed from the WORKSPACE directly.)
13. When a distribution **D** is removed from the WORKSPACE, the we will remove all the files in **D** from the WORKSPACE tree. If a file has been modified from the original as installed from the distribution (per md5sum) then the user should be asked if he is "sure" he wants to remove it.
14. When a distribution is created, a GUID is generated and assigned to the distribution. If a distribution is created from the same components at a later time, it should have the same GUID.
15. If a package **P** is marked with **`P->RePackage==false`**, then **P** may not be added to a distribution.
16. A distribution **D** is identical to a distribution G, if and only if **`D->GUID == G->GUID and D->Version == G->Version`**.
17. A distribution **D** may be installed into the WORKSPACE **W**, if and only if there is no distribution **G** in **W** such that **`D->GUID == G->GUID and D->Version == G->Version`**.

4
Distribution Description XML Schema

The Distribution Description XML Schema defines the contents of a distribution surface area including all associated meta-data. This chapter defines the XML Grammar that is used to describe the contents of the distribution package. This distribution surface area description data resides in the Distribution Description File.

The XML Schema Namespace is TBD (most likely, http://www.uefi.org/2008/2.1 or similar.)

This and following chapters of the document describe the schema in detail.

The XML Instance Representation for DistributionPackage is as follows:

```
<?xml version="1.0"?>
<DistributionPackage xmlns="http://www.uefi.org/2008/2.1"
xmlns:xsi="http:/www.w3.org/2001/XMLSchema-instance" >
  <DistributionHeader> ... </DistributionHeader> {1}
  <PackageSurfaceArea> ... </PackageSurfaceArea> {0,}
  <ModuleSurfaceArea> ... </ModuleSurfaceArea> {0,}
  <Tools>  ... </Tools> {0,}
  <MiscellaneousFiles> ... </MiscellaneousFiles> {0,}
  <UserExtensions> ... </UserExtensions> {0,}
</DistributionPackage>
```

Table 10. DistributionPackage

Description	This is the root element of the distribution XML schema.
Required	YES
Data Type	Element – Complex
Data Constraints	N/A
Examples	N/A

4.1 DistributionPackage.DistributionHeader

The following is the description of the **DistributionPackage.DistributionHeader** instance:

The following is the description of the **DistributionPackage.DistributionHeader** instance:

```
<DistributionHeader
  ReadOnly=" xs:boolean " {0,1}
  RePackage=" xs:boolean " {0,1} >
    <Name
      BaseName=" xs:NCName " {0,1} >
        xs:normalizedString
    </Name> {1}
    <GUID
      Version=" xs:decimal " {1} >
        RegistryFormatGUID
    </GUID> {1}
    <Vendor> xs:normalizedString </Vendor> {1}
    <Date> xs:dateTime </Date> {1}
    <Copyright> xs:string </Copyright> {1}
    <License> xs:string </License> {1}
    <Abstract> xs:normalizedString </Abstract> {1}
    <Description> xs:string </Description> {0,1}
    <Signature> Md5Sum </Signature> {0,1}
    <XmlSpecification> xs:decimal </XmlSpecification> {1}
</DistributionHeader>
```

Table 11. DistributionPackage.DistributionHeader

Description	This is the Header Meta data for the distribution. Most of the content is required by legal departments.
Required	YES
Data Type	Element – Complex
Data Constraints	N/A
Examples	N/A

4.1.1 DistributionHeader:ReadOnly

Table 12. DistributionHeader:ReadOnly

Description	If set to true, all content within this distribution archive should NOT be modified.
Required	NO
Data Type	Attribute – xs:boolean
Data Constraints	Either "**true**" or "**false**" (default)
Examples	**ReadOnly="true"**

4.1.2 DistributionHeader:RePackage

Table 13. DistributionHeader:RePackage

Description	If set to true, then the distribution and its content may be packaged into another distribution.
Required	NO
Data Type	Attribute – xs:boolean
Data Constraints	Either "`true`" or "`false`" (default)
Examples	`RePackage="true"`

4.1.3 DistributionHeader.Name

Table 14. DistributionHeader.Name

Description	The User Interface Name of the Distribution Package File.
Required	YES
Data Type	Element – xs:normalizedString
Data Constraints	String value of more than one character. Limited to ASCII printable characters between 0x20 and 0x7f, inclusive.
Examples	`<Name> ICH7 Chipset </Name>`

4.1.4 DistributionHeader.Name:BaseName

Table 15. DistributionHeader.Name:BaseName

Description	The reference name of the Distribution Package File. This single word name can be used by tools as a keyword or for directory and/or file creation.
Required	NO
Data Type	Attribute – xs:NCName
Data Constraints	String value of more than one character. No white space characters are permitted.
Examples	`Name="P_882"` `Name="MdePkg"`

4.1.5 DistributionHeader.GUID

Table 16. DistributionHeader.GUID

Description	The 128 bit Unique ID of the package. This is used to check if the distribution is already installed. This value must change if the entire distribution archive does not provide backward compatibility.
Required	YES
Data Type	Element – RegistryFormatGuid
Data Constraints	A string that represents a GUID in registry format.

Examples	`<GUID Version="1.2"> AF0DDA2E-EA83-480b-B2CE-FC0BB2F894C2 </GUID>`

4.1.6 DistributionHeader.GUID:Version

Table 17. DistributionHeader.GUID:Version

Description	The version of the distribution. It is assumed that the version is used to give information to the user about the distribution, but the tools will use the version of the distribution to compare to the version of another distribution with the same GUID. If the distribution contains backward compatible content, then only this version number will change. The higher the number, the more recent the content.
Required	YES
Data Type	Attribute – xs:decimal
Data Constraints	A decimal value that must include the period character.
Examples	`Version="1.2"`

4.1.7 DistributionHeader.Vendor

Table 18. DistributionHeader.Vendor

Description	A string identifying who created this distribution package.
Required	YES
Data Type	Element – xs:normalizedString
Data Constraints	Any string
Examples	`<Vendor> Nosuch Corporation </Vendor>`

4.1.8 DistributionHeader.Date

Table 19. DistributionHeader.Date

Description	The date and time that this distribution was created.
Required	YES
Data Type	Element – xs:dateTime
Data Constraints	YYYY-MM-DDThh:mm:ss[.uuuuu], the 'T' character is used to separate the date from the time, the microsecond entry is optional.
Examples	`<Date> 2008-01-01T13:30:44 </Date>` `<Date>2008-03-24T09:30:00</Date>`

4.1.9 DistributionHeader.Copyright

Table 20. DistributionHeader.Copyright

Description	The copyright for this file that is generated by the creator of the distribution. If a derivative work is generated from an existing distribution, then the existing copyright must be maintained, and additional copyrights may be appended to the end of this element. It may also be the primary copyright for all code provided in the distribution.

Required	YES
Data Type	Element – xs:string
Data Constraints	One or more lines of text.
Examples	**<Copyright>** **Copyright (c) 2008, Nosuch Corporation. All rights reserved.** **</Copyright>**

4.1.10 DistributionHeader.License

Table 21. DistributionHeader.License

Description	A license that describes any restrictions on the use of this distribution. If a derivative work is allowed by the original license and a derivative work is generated from an existing distribution, then the existing license must be maintained, and additional licenses may be appended to the end of this element. It may also be the primary license for all code provided in the Distribution Package File. Alternatively, this may point to a filename that contains the License. The file will be stored in the same location as the Distribution Description File.
Required	YES
Data Type	Element – xs:string
Data Constraints	Paragraph or file name. If the filename, then the file must have a file extension of either .txt or .lic, otherwise any number of lines of text.
Examples	**<License>** **This program and the accompanying materials are licensed and made available under the terms and conditions of the BSD License which accompanies this distribution. The full text of the license may be found at:** **http://opensource.org/licenses/bsd-license.php** **THE PROGRAM IS DISTRIBUTED UNDER THE BSD LICENSE ON AN AS IS BASIS, WITHOUT WARRANTIES OR REPRESENTATIONS OF ANY KIND, EITHER EXPRESS OR IMPLIED.** **</License>** **<License> MyLicense.txt </License>** **<License> NoSuchLicense.lic </License>**

4.1.11 DistributionHeader.Abstract

Table 22. DistributionHeader.Abstract

Description	A one line description of the distribution.
Required	YES
Data Type	Element – xs:normalizedString
Data Constraints	N/A
Examples	**<Abstract> Version 1 of the NoSuchPkg package. </Abstract>**

4.1.12 DistributionHeader.Description

Table 23. DistributionHeader.Description

Description	A complete description of the distribution. This description may include the release name of the file, the version of the file, and a complete description of the file contents and/or features including a description of the updates since the previous file release.
Required	NO
Data Type	Element – xs:string
Data Constraints	Multi-line description.
Examples	`<Description> Initial release of the NoSuchPkg, Version 1.0 providing the base data types and libraries for NoSuch UEFI Driver. This includes special libraries for reporting status codes of the driver. </Description>`

4.1.13 DistributionHeader.Signature

Table 24. DistributionHeader.Signature

Description	MD5sum of the Distribution Content File included in this Distribution Package File.
Required	NO
Data Type	Element – Md5Sum
Data Constraints	A 128-bit value, specified in hex format (32 characters.) A leading 0x is not permitted.
Examples	`<Signature> 09fa8fc7222da9afd9ffd52ba8b73f45 </ Signature>`

4.1.14 DistributionHeader.XmlSpecification

Table 25. DistributionHeader.XmlSpecification

Description	The XML Schema Specification to which this Distribution Description conforms.
Required	YES
Data Type	Element – xs:decimal
Data Constraints	Distribution Description XML Schema version. For this version, the value is 1.0.
Examples	`<XmlSpecification> 1.0 </XmlSpecification>`

5
Package Surface Area Description

This section describes the Package Surface Area section of the Distribution Description File. Zero or more of these sections are permitted within the Distribution Description File.

The XML Instance Representation for **DistributionPackage.PackageSurfaceArea** is as follows:

```
<PackageSurfaceArea>
    <Header> ... </Header> {1}
    <ClonedFrom> ... </ClonedFrom> {0,1}
    <LibraryClassDeclarations> ... </LibraryClassDeclarations>
{0,1}
    <IndustryStandardIncludes> ... </IndustryStandardIncludes>
{0,1}
    <PackageIncludes> ... </PackageIncludes> {0,1}
    <Modules> ... </Modules> {0,1}
    <GuidDeclarations> ... </GuidDeclarations> {0,1}
    <ProtocolDeclarations> ... </ProtocolDeclarations> {0,1}
    <PpiDeclarations> ... </PpiDeclarations> {0,1}
    <PcdDeclarations> ... </PcdDeclarations> {0,1}
    <PcdRelationshipChecks> ... </PcdRelationshipChecks> {0,1}
    <MiscellaneousFiles> ... </MiscellaneousFiles> {0,1}
    <UserExtensions> ... </UserExtensions> {0,1}
</PackageSurfaceArea>
```

Table 26. DistributionPackage.PackageSurfaceArea

Description	This element contains all the information associated with a Package Surface Area
Required	NO
Data Type	Element – Complex
Data Constraints	The Header element is required. All other elements are optional.
Examples	N/A

5.1 PackageSurfaceArea.Header

The following is the description of the **PackageSurfaceArea.Header** instance:

```
<Header>
  <Name
    BaseName=" xs:NCName " {1} >
      xs:normalizedString
  </Name> {1}
  <GUID
    Version=" xs:decimal " {1} >
      RegistryFormatGuid
  </GUID> {1}
  <Copyright> xs:string </Copyright> {0,1}
  <License> xs:string </License> {0,1}
  <Abstract> xs:normalizedString </Abstract> {0,1}
  <Description> xs:string </Description> {0,1}
  <PackagePath> xs:anyURI </PackagePath> {1}
</Header>
```

Table 27. PackageSurfaceArea.Header

Description	This element contains the header information for a Package Surface Area. This includes the name of the package, copyright and licensing information associated with the package.
Required	YES
Data Type	Element – Complex
Data Constraints	N/A
Examples	N/A

5.1.1 Header.Name

Table 28. Header.Name

Description	The User Interface Name for the package.
Required	YES
Data Type	Element – xs:normalizedString
Data Constraints	String value of more than one character.
Examples	`<Name>MDE Package Version 1.00</Name>` `<Name>NT32 Package Version 2.76</PackageName>`

5.1.2 Header.Name:BaseName

Table 29. Header.Name:BaseName

Description	The reference name of the package. This single word name can be used by tools as a keyword and for directory/file creation.
Required	YES

Data Type	Attribute – xs:NCName
Data Constraints	String value of more than one character. No white space characters are permitted.
Examples	**Name="P_882"** **Name="MdePkg"**

5.1.3 Header.GUID

Table 30. Header.GUID

Description	The 128-bit GUID VALUE that is the unique name of a package. This value must change any time a release of the package does not provide full backward compatibility.
Required	YES
Data Type	Element – RegistryFormatGuid
Data Constraints	A string that represents a GUID in registry format. If a package is backwards compatible with a previous release of the same package, then **PackageSurfaceArea.Header.GUID** element must not be changed, and only the **PackageSurfaceArea.Header.GUID:Version** attribute should be increased. If a package is not backward compatible with a previous release or a new package is being created, then a new **PackageSurfaceArea.Header.GUID** must be generated.
Examples	**<GUID> AF0DDA2E-EA83-480b-B2CE-FC0BB2F894C2 </GUID>**

5.1.4 Header.GUID:Version

Table 31. Header.GUID:Version

Description	The version of this package. The version number must change (incrementally) for every release of a package. Since it is tied to the package GUID, this value may be reset to an initial number, if the GUID changes due to a non-backward compatible release.
Required	YES
Data Type	Attribute – xs:decimal
Data Constraints	A string that contains a decimal number with a dot separator.
Examples	**Version="1.00"** **Version="1.02"** **Version="3.27"**

5.1.5 Header.Copyright

Table 32. Header.Copyright

Description	The copyright for this package if it is different than the copyright of the distribution package. The copyright is generated by the creator of a package. If a derivative work is generated from an existing package, then the existing copyright must be maintained, and additional copyrights may be appended to the end of this element.
Required	NO
Data Type	Element – xs:string
Data Constraints	A set of one or more copyright statements on one or more lines of text.

Examples	`<Copyright> Copyright (c) 2006, Nosuch Corporation. All rights reserved. </Copyright>`

5.1.6 Header.License

Table 33. Header.License

Description	A license that describes any restrictions on the use of this package. If a derivative work is allowed by the original license and a derivative work is generated from an existing package, then the existing license must be maintained, and additional licenses may be appended to the end of this element. Alternatively, this may point to a file that will be in the same folder as the package declaration file.
Required	NO
Data Type	Element – xs:string
Data Constraints	A string that contains one or more lines of text. If a file, then the file must be an ASCII text file with either a .txt or .lic extension.
Examples	**<License> This program and the accompanying materials are licensed and made available under the terms and conditions of the BSD License which accompanies this distribution. The full text of the license may be found at http://opensource.org/licenses/bsd-license.php THE PROGRAM IS DISTRIBUTED UNDER THE BSD LICENSE ON AN AS IS BASIS, WITHOUT WARRANTIES OR REPRESENTATIONS OF ANY KIND, EITHER EXPRESS OR IMPLIED. </License> <License> License.txt </License>**

5.1.7 Header.Abstract

Table 34. Header.Abstract

Description	A one line description of this package if different from the distribution's abstract.
Required	NO
Data Type	Element – xs:normalizedString
Data Constraints	N/A
Examples	**<Abstract>Headers for NoSuch Development Environment Core (MDE.)</Abstract>**

5.1.8 Header.Description

Table 35. Header.Description

Description	A complete description of a package. This description may include the release name of the package, the version of the package, and a complete description of the package contents and/or features including a description of the updates since the previous package's release.
Required	NO
Data Type	Element – xs:string
Data Constraints	A string that contains one or more lines of text.
Examples	`<Description> NoSuch Package Version 0.75 that conforms to the NoSuch Package Specification Version 0.75. This package contains all the include files, Protocols, PPIs, GUIDs, Library Classes, and Library instances required to build modules that are compliant with the UEFI/PI Specifications. This package was updated from Version 0.74 and now includes addition Base Library String Functions. </Description>`

5.1.9 Header.PackagePath

Table 36. Header.PackagePath

Description	This is the location (in the Distribution Content File) for the root directory of a package.
Required	YES
Data Type	Element – xs:anyURI
Data Constraints	A directory location (in the Distribution Content File.)
Examples	`<PackagePath> NoSuchPkg </PackagePath>`

5.2 PackageSurfaceArea.ClonedFrom

This section is used to track a package that was cloned (copied from) an existing package.

The following is the description of the `PackageSurfaceArea.ClonedFrom` instance:

```
<ClonedFrom>
  <GUID
    Version=" xs:decimal " {1} >
      RegistryFormatGuid
  </GUID> {1}
</ClonedFrom>
```

Table 37. PackageSurfaceArea.ClonedFrom

Description	If the package sources were copied from an existing package, then this Section can be used to track the lineage.
Required	NO
Data Type	Element – Complex
Data Constraints	N/A
Examples	N/A

5.2.1 ClonedFrom.GUID

Table 38. ClonedFrom.GUID

Description	If the package sources were copied from an existing package, this is the GUID of the package that was the source of the copy.
Required	YES
Data Type	Element – RegistryFormatGuid
Data Constraints	Registry format GUID of the pre-existing package
Examples	`<GUID Version="1.0"> 27d67720-ea68-48ae-93da-a3a074c90e30 </GUID>`

5.2.2 ClonedFrom.GUID:Version

Table 39. ClonedFrom.GUID:Version

Description	If the package sources were copied from an existing package, this is the version of the package that was the source of the copy.
Required	YES
Data Type	Attribute – xs:decimal
Data Constraints	Version number of the pre-existing package
Examples	`Version="1.0"`

5.3 PackageSurfaceArea.LibraryClassDeclarations

This section provides the mapping of Library Class keywords to the header files that define the interfaces and data structures that a library class requires. All Library Class declarations must be unique to the package.

The following is the description of the **PackageSurfaceArea.LibraryClassDeclarations** instance:

```
<LibraryClassDeclarations>
  <LibraryClass
    Keyword=" xs:NCName " {1}
    SupArchList=" ArchListType " {0,1}
    SupModList=" ModuleListType " {0,1} >
      <HeaderFile> xs:anyURI </HeaderFile> {1}
      <RecommendedInstance>
          <GUID
            Version=" xs:decimal " {0,1} >
              RegistryFormatGuid
          </GUID> {1}
      </RecommendedInstance> {0,1}
      <HelpText
         Lang=" xs:language " {0,1} >
           xs:string
      </HelpText> {0,}
  </LibraryClass> {1,}
</LibraryClassDeclarations>
```

Table 40. PackageSurfaceArea.LibraryClassDeclarations

Description	The list of library classes that the package declares.
Required	NO
Data Type	Element – Complex
Data Constraints	N/A
Examples	N/A

5.3.1 LibraryClassDeclarations.LibraryClass

Table 41. LibraryClassDeclarations.LibraryClass

Description	The individual Library Class declared – a keyword and a header file.
Required	YES
Data Type	Element – Complex
Data Constraints	N/A
Examples	N/A

5.3.2 LibraryClassDeclarations.LibraryClass:Keyword

Table 42. LibraryClassDeclarations.LibraryClass:Keyword

Description	Modules may be coded to use a library class header identified by this keyword.
Required	YES
Data Type	Attribute – xs:NCName
Data Constraints	A string that start with a letter or an underscore and is followed by any combination of letters, digits, periods, hyphens, and underscores. No whitespace is allowed.
Examples	**Keyword="BaseLib"** **Keyword="PrintLib"**

5.3.3 LibraryClassDeclarations.LibraryClass:SupArchList

Table 43. LibraryClassDeclarations.LibraryClass:SupArchList

Description	Used to restrict the set of CPU Architectures that are allowed to use this Library Class. If this attribute is not specified, then this Library Class may be used with any CPU Architecture. If this attribute is specified, then only those modules that support a subset of the CPU architectures specified by this element may use this Library Class.
Required	NO
Data Type	Attribute – ArchListType
Data Constraints	If specified, must contain one or more of the supported CPU architectures separated by spaces. The supported CPU architectures include **IA32**, **X64**, **IPF** and **EBC**

Examples	**SupArchList=”IA32”** **SupArchList=”IA32 X64”** **SupArchList=”EBC IPF X64”**

5.3.4 LibraryClassDeclarations.LibraryClass:SupModList

Table 44. LibraryClassDeclarations.LibraryClass:SupModList

Description	Used to restrict the set of module types that are allowed to use this Library Class. If this attribute is not specified, then this Library Class may be used with all module types. If this attribute is specified, then only those modules that have a module type that is a member of the set of module types specified by this element may use this Library Class.
Required	NO
Data Type	Attribute – ModuleListType
Data Constraints	If specified, must contain one or more of the supported module types separated by spaces. The supported module types include **BASE**, **SEC**, **PEI_CORE**, **PEIM**, **DXE_CORE**, **DXE_DRIVER**, **DXE_RUNTIME_DRIVER**, **DXE_SAL_DRIVER**, **DXE_SMM_DRIVER**, **TOOL**, **UEFI_DRIVER**, **UEFI_RUNTIME_DRIVER**, **UEFI_APPLICATION**, **USER_DEFINED**.
Examples	**SupModList="BASE"** **SupModList="PEIM DXE_DRIVER"** **SupModList="UEFI_DRIVER UEFI_APPLICATION DXE_DRIVER"**

5.3.5 LibraryClassDeclarations.LibraryClass.HeaderFile

Table 45. LibraryClassDeclarations.LibraryClass.HeaderFile

Description	This is element is a single header file that defines the library class calling conventions and format.
Required	YES
Data Type	Element – xs:anyURI
Data Constraints	The PackagePath relative path and filename in the Distribution Content Content File for the include file associated with the Library Class.
Examples	**<HeaderFile>Include/Library/PeiServicesLib.h</HeaderFile>** **<HeaderFile> Include/Library/HobLib.h </HeaderFile>**

5.3.6 LibraryClassDeclarations.LibraryClass.RecommendedInstance

Table 46. LibraryClassDeclarations.LibraryClass.RecommendedInstance

Description	The recommended library instance as defined by the creator of the Library Class Header file.
Required	NO
Data Type	Element – Complex
Data Constraints	N/A

Examples	N/A

5.3.7 LibraryClassDeclarations.LibraryClass.RecommendedInstance.GUID

Table 47. LibraryClassDeclarations.LibraryClass.RecommendedInstance.GUID

Description	The GUID of the recommended library instance as defined by the creator of the Library Class Header file. This is a required element. If GUID is specified, and the Version attribute is not, then the algorithm to look up the recommended instance is to find the library instance highest version number and a matching GUID.
Required	YES
Data Type	Element – RegistryFormatGuid
Data Constraints	A string that represents a GUID in registry format. Not valid if no RecommendedInf attribute specified.
Examples	`<GUID>AF0DDA2E-EA83-480b-B2CE-FC0BB2F894C2</GUID>`

5.3.8 LibraryClassDeclarations.LibraryClass.RecommendedInstance.GUID:Version

Table 48. LibraryClassDeclarations.LibraryClass:RecommendedInstance.GUID:Version

Description	The version of the recommended library instance as defined by the creator of the Library Class Header file. If this attribute is not specified, then the recommended library instance is the highest version number with a matching GUID.
Required	NO
Data Type	Attribute – xs:decimal
Data Constraints	Unsigned decimal number.
Examples	`Version="1.0"` `Version="2.75"`

5.3.9 LibraryClassDeclarations.LibraryClass.HelpText

Table 49. LibraryClassDeclarations.LibraryClass.HelpText

Description	A paragraph describing this library class, including usage models and restrictions.
Required	NO
Data Type	Element – xs:string
Data Constraints	A string that contains one or more lines of text.
Examples	`<HelpText> This is the include file for any module of type BASE. Base modules only use types defined via this include file and can be ported easily to any environment. There are a set of base libraries in the NoSuch Package that can be used to implement base modules. </HelpText>`

5.3.10 LibraryClassDeclarations.LibraryClass.HelpText:Lang

Table 50. LibraryClassDeclarations.LibraryClass.HelpText:Lang

Description	The language code uses an RFC 1766 language code identifier
Required	NO
Data Type	Attribute – xs:language
Data Constraints	If present, the language code must use an RFC 1766 language code identifier. If not specified, the default is “en-us.’
Examples	`Lang=”en-us”`

5.4 PackageSurfaceArea.IndustryStandardIncludes

The following is the description of the `PackageSurfaceArea.IndustryStandardIncludes` instance:

```
<IndustryStandardIncludes>
  <IndustryStandardHeader>
    <HeaderFile> xs:anyURI </HeaderFile> {1}
    <HelpText
       Lang=" xs:language " {0,1} >
         xs:string
    </HelpText> {0,}
  </IndustryStandardHeader> {1,}
</IndustryStandardIncludes>
```

Table 51. PackageSurfaceArea.IndustryStandardIncludes

Description	This is element is a single industry standard header .
Required	YES
Data Type	Element – Complex
Data Constraints	Must contain one and only one Header File
Examples	N/A

5.4.1 IndustryStandardIncludes.IndustryStandardHeader

Table 52. IndustryStandardIncludes.IndustryStandardHeader

Description	This is element is a single industry standard header .
Required	YES
Data Type	Element – Complex
Data Constraints	Must contain one and only one Header File
Examples	N/A

5.4.2 IndustryStandardIncludes.IndustryStandardHeader.HeaderFile

Table 53. IndustryStandardIncludes.IndustryStandardHeader.HeaderFile

Description	This is element is a single industry standard header file that maybe used by multiple modules.
Required	YES
Data Type	Element – xs:anyURI
Data Constraints	A package relative path and filename (as located in the Distribution Content File) for the include file associated with the Industry standard header file.
Examples	`<HeaderFile>Include/IndustryStandard/Pci.h</ HeaderFile>` `<HeaderFile> Include/IndustryStandard/Acpi.h </ HeaderFile>`

5.4.3 IndustryStandardIncludes.IndustryStandardHeader.HelpText

Table 54. IndustryStandardIncludes.IndustryStandardHeader.HelpText

Description	A paragraph describing this header file, including usage models and restrictions.
Required	NO
Data Type	Element – xs:string
Data Constraints	A string that contains one or more lines of text.
Examples	`<HelpText USC2="false" Lang="en-us"> This file contains the latest ACPI definitions that are consumed by drivers that do not care about ACPI versions.</HelpText>`

5.4.4 IndustryStandardIncludes.IndustryStandardHeader.HelpText:Lang

Table 55. IndustryStandardIncludes.IndustryStandardHeader.HelpText:Lang

Description	The language code uses an RFC 1766 language code identifier
Required	NO
Data Type	Attribute – xs:language
Data Constraints	If present, the language code must use an RFC 1766 language code identifier. If not specified, the default is “en-us.’
Examples	`Lang="en-us"`

5.5 PackageSurfaceArea.PackageIncludes

This section is use to list header files that are a subset of other header files specified for either GUIDs, Protocols and/or PPIs or for supporting a given module type.

The following is the description of the `PackageSurfaceArea.PackageIncludes` instance:

```
<PackageIncludes>
  <PackageHeader>
    <HeaderFile
      SupArchList=" ArchListType " {0,1}
      SupModList=" ModuleListType " {0,1} >
        xs:anyURI
    </HeaderFile> {1}
    <HelpText
       Lang=" xs:language " {0,1} >
         xs:string
    </HelpText> {0,}
  </PackageHeader> {1,}
</PackageIncludes>
```

Table 56. PackageSurfaceArea.PackageIncludes

Description	Specifies the package relative filename of an include file that extends the set of definitions available for the standard module types.
Required	NO
Data Type	Element – Complex
Data Constraints	N/A
Examples	N/A

5.5.1 PackageIncludes.PackageHeader

Table 57. PackageIncludes.PackageHeader

Description	Specifies the package declaration relative filename of an include file that extends the set of definitions available for the standard module types.
Required	YES
Data Type	Element – Complex
Data Constraints	Do not list: headers that are local to a module (module specific,) industry standard headers or a library class header in this section.
Examples	N/A

5.5.2 PackageIncludes.PackageHeader.HeaderFile

Table 58. PackageIncludes.PackageHeader.HeaderFile

Description	Specifies the package declaration relative filename of an include file that extends the set of definitions available for the standard module types.
Required	YES
Data Type	Element – xs:anyURI
Data Constraints	A PackagePath relative Path and Filename (as located in the Distribution Content File.)

Examples	`<HeaderFile>Include/Peim.h</HeaderFile>` `<HeaderFile> Include/EdkDxeCore.h </HeaderFile>`

5.5.3 PackageIncludes.PackageHeader.HeaderFile:SupArchList

Table 59. PackageIncludes.PackageHeader.HeaderFile:SupArchList

Description	Used to restrict the set of CPU Architectures that are allowed to use this header. If this attribute is not specified, then this header may be used with any CPU Architecture. If this attribute is specified, then only those modules that support a subset of the CPU architectures specified by this element may use header.
Required	NO
Data Type	Attribute – ArchListType
Data Constraints	If specified, must contain one or more of the supported CPU architectures separated by spaces. The supported CPU architectures include **IA32**, **X64**, **IPF** and **EBC**
Examples	**SupArchList="IA32"** **SupArchList="IA32 X64"** **SupArchList="EBC IPF X64"**

5.5.4 PackageIncludes.PackageHeader.HeaderFile:SupModList

Table 60. PackageIncludes.PackageHeader.HeaderFile:SupModList

Description	Used to restrict the set of module types that are allowed to use this header. If this attribute is not specified, then this header may be used with all module types. If this attribute is specified, then only those modules that have a module type that is a member of the set of module types specified by this element may use this header.
Required	NO
Data Type	Attribute – ModuleListType
Data Constraints	A string that contains a list of one or more module types. Module types are restricted to one of **BASE**, **SEC**, **PEI_CORE**, **PEIM**, **DXE_CORE**, **DXE_DRIVER**, **DXE_RUNTIME_DRIVER**, **DXE_SAL_DRIVER**, **DXE_SMM_DRIVER**, **TOOL**, **UEFI_DRIVER**, **UEFI_RUNTIME_DRIVER**, **UEFI_APPLICATION**, **USER_DEFINED**.
Examples	**SupModList="BASE"** **SupModList="PEI_CORE PEIM"** **ModuleType="DXE_CORE UEFI_DRIVER DXE_DRIVER"**

5.5.5 PackageIncludes.PackageHeader.HelpText

Table 61. PackageIncludes.PackageHeader.HelpText

Description	A paragraph describing this header file, including usage models and restrictions.
Required	NO
Data Type	Element – xs:string
Data Constraints	A string that contains one or more lines of text.

Examples	`<HelpText USC2="false" Lang="en-us"> This file contains the latest ACPI definitions that are consumed by drivers that do not care about ACPI versions.</HelpText>`

5.5.6 PackageIncludes.PackageHeader.HelpText:Lang

Table 62. PackageIncludes.PackageHeader.HelpText:Lang

Description	The language code uses an RFC 1766 language code identifier
Required	NO
Data Type	Attribute – xs:language
Data Constraints	If present, the language code must use an RFC 1766 language code identifier. If not specified, the default is "en-us.'
Examples	`Lang="en-us"`

5.6 PackageSurfaceArea.Modules

This section is used to reference individual modules that are part of a package.

The following is the description of the **PackageSurfaceArea.Module**s instance:

```
<Modules>
  <ModuleSurfaceArea> ... </ModuleSurfaceArea> {1,}
</Modules>
```

Table 63. PackageSurfaceArea.Modules

Description	This is section may be used to include a list of modules that are part of this package. An module listed here must have a corresponding ModuleSurfaceArea section completed.
Required	NO
Data Type	Element – Complex
Data Constraints	N/A
Examples	N/A

5.6.1 Modules.ModuleSurfaceArea

Refer to chapter 6, Module Surface Area Description. Modules that are part of a package should be listed here. Stand-alone modules that are not part of any package should be listed under the top level ModuleSurfaceArea section of the DistributionPackage.

5.7 PackageSurfaceArea.GuidDeclarations

This section defines all non-Protocol and non-Ppi C names and GUID values declared by a package. It is only specified in the package that declares the GUID in the design document.

The following is the description of the **PackageSurfaceArea.GuidDeclarations** instance:

```
<GuidDeclarations>
  <Entry
    UiName=" xs:normalizedString " {0,1}
    GuidTypes=" GuidListType " {0,1}
    SupArchList=" ArchListType " {0,1}
    SupModList=" ModuleListType " {0,1} >
      <CName> xs:NCName </CName> {1}
      <GuidValue> RegistryFormatGuid </GuidValue> {1}
      <HelpText
         Lang=" xs:language " {0,1} >
           xs:string
      </HelpText> {0,}
  </Entry> {1,}
</GuidDeclarations>
```

Table 64. PackageSurfaceArea.GuidDeclarations

Description	Specifies the list of GUIDs produced by this Package
Required	NO
Data Type	Element – Complex
Data Constraints	N/A
Examples	N/A

5.7.1 GuidDeclarations.Entry

Table 65. GuidDeclarations.Entry

Description	Specifies the individual GUID information for a GUID that is declared by a specification for this Package
Required	YES
Data Type	Element – Complex
Data Constraints	N/A
Examples	N/A

5.7.2 GuidDeclarations.Entry:UiName

Table 66. GuidDeclarations.Entry:UiName

Description	A user interface name for this GUID entry.
Required	NO
Data Type	Attribute – xs:normalizedString
Data Constraints	A string that start with a letter or an underscore and is followed by any combination of letters, digits, periods, hyphens, and underscores. No whitespace is allowed.
Examples	**UiName="CustomDecompress"**

5.7.3 GuidDeclarations.Entry:GuidTypes

Table 67. GuidDeclarations.Entry:GuidTypes

Description	Used to restrict the set of GUID types that apply to this GUID. If this attribute is not specified, then this GUID may be used with all GUID types. If this attribute is specified, then this GUID may only appear in the sections of the module's information (build implementation specific) file that have a matching GUID type.
Required	NO
Data Type	Attribute – GuidListType
Data Constraints	A string that contains one or more GUID types separated by spaces. The GUID types are restricted to **Event**, **File**, **FV**, **HII**, **HOB**, **SystemTable**, **TokenSpaceGuid**, **Variable** or **GUID** (default.)
Examples	**GuidTypes="TokenSpaceGuid"** **GuidTypes="SystemTable"** **GuidTypes="HOB SystemTable"**

5.7.4 GuidDeclarations.Entry:SupArchList

Table 68. GuidDeclarations.Entry:SupArchList

Description	Used to restrict the set of CPU Architectures that are allowed to use this GUID. If this attribute is not specified, then this GUID may be used with any CPU Architecture. If this attribute is specified, then only those modules that support a subset of the CPU architectures specified by this element may use this GUID.
Required	NO
Data Type	Attribute – ArchListType
Data Constraints	If specified, must contain one or more of the supported CPU architectures separated by spaces. The supported CPU architectures include **IA32**, **X64**, **IPF** and **EBC**
Examples	**SupArchList="IA32"** **SupArchList="IA32 X64"** **SupArchList="EBC IPF X64"**

5.7.5 GuidDeclarations.Entry:SupModList

Table 69. GuidDeclarations.Entry:SupModList

Description	Used to restrict the set of module types that are allowed to use this GUID. If this attribute is not specified, then this GUID may be used with all module types. If this attribute is specified, then only those modules that have a module type that is a member of the set of module types specified by this element may use this GUID.
Required	NO
Data Type	Attribute – ModuleListType
Data Constraints	If specified, must contain one or more of the supported module types separated by spaces. The supported module types include **BASE**, **SEC**, **PEI_CORE**, **PEIM**, **DXE_CORE**, **DXE_DRIVER**, **DXE_RUNTIME_DRIVER**, **DXE_SAL_DRIVER**, **DXE_SMM_DRIVER**, **TOOL**, **UEFI_DRIVER**, **UEFI_RUNTIME_DRIVER**, **UEFI_APPLICATION**, **USER_DEFINED**.
Examples	**SupModList="BASE"** **SupModList="PEIM DXE_DRIVER"** **SupModList="UEFI_DRIVER UEFI_APPLICATION DXE_DRIVER"**

5.7.6 GuidDeclarations.Entry.CName

Table 70. GuidDeclarations.Entry.CName

Description	Specifies the symbol name for a GUID used in C code.
Required	YES
Data Type	Element – xs:NCName
Data Constraints	A valid C variable name string that starts with either an underscore or a letter, followed by any number of letters, digits or underscores.
Examples	**<CName>Acpi10Table</CName>** **<CName>PcAnsi</CName>** **<CName>DxeServicesTable</CName>**

5.7.7 GuidDeclarations.Entry.GuidValue

Table 71. GuidDeclarations.Entry.GuidValue

Description	Specifies the registry format value of the GUID.
Required	YES
Data Type	Element – RegistryFormatGuid
Data Constraints	A registry format GUID.

Examples	**<GuidValue> 8BE4DF61-93CA-11D2-AA0D-00E098032B8C </GuidValue>** **<GuidValue>DFA66065-B419-11D3-9A2D-0090273FC14D</GuidValue>**

5.7.8 GuidDeclarations.Entry.HelpText

Table 72. GuidDeclarations.Entry.HelpText

Description	A complete description of a GUID. This must include any defines and data structures associated with this GUID. It must also describe any use restrictions based on GUID Type, Module Type, CPU Architecture, and/or Feature Flags.
Required	NO
Data Type	Element – xs:string
Data Constraints	A string that contains one or more lines of text.
Examples	**<HelpText Lang="en-us">The Console In Device GUID is used to tag an EFI_HANDLE for a Console Input Device as an active console input device. This is used by the Console Splitter Driver to search for set of console input devices in the handle database that should be multiplexed. This GUID is available to all CPU types, but is restricted for use by DXE_DRIVER, UEFI_DRIVER, and UEFI_APPLICATION module types.</HelpText>**

5.7.9 GuidDeclarations.Entry.HelpText:Lang

Table 73. GuidDeclarations.Entry.HelpText:Lang

Description	The language code uses an RFC 1766 language code identifier
Required	NO
Data Type	Attribute – xs:language
Data Constraints	If present, the language code must use an RFC 1766 language code identifier. If not specified, the default is "en-us.'
Examples	**Lang="en-us"**

5.8 PackageSurfaceArea.ProtocolDeclarations

This section defines all Protocol and ProtocolNotify C names and the GUID values declared by a package. It is only specified in the package that declares the Protocol in the design document.

The following is the description of the **PackageSurfaceArea.ProtocolDeclarations** instance:

```
<ProtocolDeclarations>
  <Entry
    UiName=" xs:normalizedString " {0,1}
    SupArchList=" ArchListType " {0,1}
    SupModList=" ModuleListType " {0,1} >
      <CName> xs:NCName </CName> {1}
      <GuidValue> RegistryFormatGuid </GuidValue> {1}
      <HelpText
         Lang=" xs:language " {0,1} >
           xs:string
      </HelpText> {0,}
  </Entry> {1,}
</ProtocolDeclarations>
```

Table 74. PackageSurfaceArea.ProtocolDeclarations

Description	Specifies the list of protocols that are produced by this package.
Required	NO
Data Type	Element – Complex
Data Constraints	N/A
Examples	N/A

5.8.1 ProtocolDeclaration.Entry

Table 75. ProtocolDeclarations.Entry

Description	Specifies the individual Protocol information about the protocol that must be produced by this package.
Required	YES
Data Type	Element – Complex
Data Constraints	N/A
Examples	N/A

5.8.2 ProtocolDeclarations.Entry:UiName

Table 76. ProtocolDeclarations.Entry:UiName

Description	A user interface name for this Protocol entry.
Required	NO
Data Type	Attribute – xs:normalizedString
Data Constraints	A string that start with a letter or an underscore and is followed by any combination of letters, digits, periods, hyphens, and underscores. No whitespace is allowed.
Examples	`UiName="IsaAcpi"` `UiName="PciHotPlugRequest"`

5.8.3 ProtocolDeclarations.Entry:SupArchList

Table 77. ProtocolDeclarations.Entry:SupArchList

Description	Used to restrict the set of CPU Architectures that are allowed to use this Protocol. If this attribute is not specified, then this Protocol may be used with any CPU Architecture. If this attribute is specified, then only those modules that support a subset of the CPU architectures specified by this element may use this Protocol.
Required	NO
Data Type	Attribute – ArchListType
Data Constraints	If specified, must contain one or more of the supported CPU architectures separated by spaces. The supported CPU architectures include **IA32**, **X64**, **IPF** and **EBC**
Examples	**SupArchList="IA32"** **SupArchList="IA32 X64"** **SupArchList="EBC IPF X64"**

5.8.4 ProtocolDeclarations.Entry:SupModList

Table 78. ProtocolDeclarations.Entry:SupModList

Description	Used to restrict the set of module types that are allowed to use this Protocol. If this attribute is not specified, then this Protocol may be used with all module types. If this attribute is specified, then only those modules that have a module type that is a member of the set of module types specified by this element may use this Protocol.
Required	NO
Data Type	Attribute – ModuleListType
Data Constraints	If specified, must contain one or more of the supported module types separated by spaces. The supported module types include **BASE**, **SEC**, **PEI_CORE**, **PEIM**, **DXE_CORE**, **DXE_DRIVER**, **DXE_RUNTIME_DRIVER**, **DXE_SAL_DRIVER**, **DXE_SMM_DRIVER**, **TOOL**, **UEFI_DRIVER**, **UEFI_RUNTIME_DRIVER**, **UEFI_APPLICATION**, **USER_DEFINED**.
Examples	**SupModList="BASE"** **SupModList="PEIM DXE_DRIVER"** **SupModList="UEFI_DRIVER UEFI_APPLICATION DXE_DRIVER"**

5.8.5 ProtocolDeclarations.Entry.CName

Table 79. ProtocolDeclarations.Entry.CName

Description	Specifies the symbol name for a Protocol GUID used in C code.
Required	YES
Data Type	Element – xs:NCName

Data Constraints	A string that starts with either an underscore or a letter, followed by any number of letters, digits or underscores.
Examples	**<CName>gEfiBlockIoProtocolGuid</CName>** **<CName> gEfiDiskIoProtocolGuid </CName>**

5.8.6 ProtocolDeclarations.Entry.GuidValue

Table 80. ProtocolDeclarations.Entry.GuidValue

Description	Specifies the GUID value for a Protocol.
Required	YES
Data Type	Element – RegistryFormatGuid
Data Constraints	The Registry Format GUID Value
Examples	`<GuidValue> AF0DDA2E-EA83-480b-B2CE-FC0BB2F894C2 </GuidValue>`

5.8.7 ProtocolDeclarations.Entry.HelpText

Table 81. ProtocolDeclarations.Entry.HelpText

Description	A complete description of a Protocol. This must include any functions, defines, and data structures associated with this Protocol. It must also describe any use restrictions based on Module Type, CPU Architecture, and/or Feature Flags.
Required	NO
Data Type	Element – xs:string
Data Constraints	A string that contains one or more lines of text.
Examples	`<HelpText>The Block I/O Protocol provides services to read, write, and flush block to a block oriented storage device such as a hard disk, CD-ROM, DVD, and floppy. This Protocol is available to all CPU types, but is restricted for use by DXE_DRIVER, UEFI_DRIVER, and UEFI_APPLICATION module types.</HelpText>`

5.8.8 ProtocolDeclarations.Entry.HelpText:Lang

Table 82. ProtocolDeclarations.Entry.HelpText:Lang

Description	The language code uses an RFC 1766 language code identifier
Required	NO
Data Type	Attribute – xs:language
Data Constraints	If present, the language code must use an RFC 1766 language code identifier. If not specified, the default is “en-us.’
Examples	`Lang=”en-us”`

5.9 PackageSurfaceArea.PpiDeclarations

This section defines all Ppi and PpiNotify C names and the GUID values declared by a package. It is

only specified in the package that declares the Ppi in the design document.

The following is the description of the **`PackageSurfaceArea.PpiDeclarations`** instance:

```
<PpiDeclarations>
  <Entry
    UiName=" xs:normalizedString " {0,1}
    SupArchList=" ArchListType " {0,1}
    SupModList=" ModuleListType " {0,1} >
      <CName> xs:NCName </CName> {1}
      <GuidValue> RegistryFormatGuid </GuidValue> {1}
      <HelpText
         Lang=" xs:language " {0,1} >
           xs:string
      </HelpText> {0,}
  </Entry> {1,}
</PpiDeclarations>
```

Table 83. PackageSurfaceArea.PpiDeclarations

Description	Specifies a list of PPIs that are produced by this Package.
Required	NO
Data Type	Element – Complex
Data Constraints	N/A
Examples	N/A

5.9.1 PpiDeclarations.Entry

Table 84. PpiDeclarations.Entry

Description	Specifies a single PPI that are produced by this Package.
Required	YES
Data Type	Element – Complex
Data Constraints	N/A
Examples	N/A

5.9.2 PpiDeclarations.Entry:UiName

Table 85. PpiDeclarations.Entry:UiName

Description	A user interface name for this PPI entry.
Required	NO
Data Type	Attribute – xs:normalizedString
Data Constraints	A string that start with a letter or an underscore and is followed by any combination of letters, digits, periods, hyphens, and underscores. No whitespace is allowed.
Examples	`UiName="BaseMemoryTest"` `UiName="OperatorPresence"`

5.9.3 PpiDeclarations.Entry:SupArchList

Table 86. PpiDeclarations.Entry:SupArchList

Description	Used to restrict the set of CPU Architectures that are allowed to use this PPI. If this attribute is not specified, then this PPI may be used with any CPU Architecture. If this attribute is specified, then only those modules that support a subset of the CPU architectures specified by this element may use this PPI.
Required	NO
Data Type	Attribute – ArchListType
Data Constraints	If specified, must contain one or more of the supported CPU architectures separated by spaces. The supported CPU architectures include **IA32**, **X64**, **IPF** and **EBC**
Examples	**SupArchList="IA32"** **SupArchList="IA32 X64"** **SupArchList="EBC IPF X64"**

5.9.4 PpiDeclarations.Entry:SupModList

Table 87. PpiDeclarations.Entry:SupModList

Description	Used to restrict the set of module types that are allowed to use this PPI. If this attribute is not specified, then this PPI may be used with all module types. If this attribute is specified, then only those modules that have a module type that is a member of the set of module types specified by this element may use this PPI.
Required	NO
Data Type	Attribute – ModuleListType
Data Constraints	If specified, must contain one or more of the supported module types separated by spaces. The supported module types include **BASE**, **SEC**, **PEI_CORE**, **PEIM**, **DXE_CORE**, **DXE_DRIVER**, **DXE_RUNTIME_DRIVER**, **DXE_SAL_DRIVER**, **DXE_SMM_DRIVER**, **TOOL**, **UEFI_DRIVER**, **UEFI_RUNTIME_DRIVER**, **UEFI_APPLICATION**, **USER_DEFINED**.
Examples	**SupModList="BASE"** **SupModList="PEIM DXE_DRIVER"** **SupModList="UEFI_DRIVER UEFI_APPLICATION DXE_DRIVER"**

5.9.5 PpiDeclarations.Entry.CName

Table 88. PpiDeclarations.Entry.CName

Description	Specifies the symbol name for a PPI GUID used in C code.
Required	YES
Data Type	Element – xs:NCName

Data Constraints	A string that starts with either an underscore or a letter, followed by any number of letters, digits or underscores.
Examples	`<CName>gEfiDxeiIplPpiGuid</CName>` `<CName> gEfiPciCfgPpiGuid </CName>`

5.9.6 PpiDeclarations.Entry.GuidValue

Table 89. PpiDeclarations.Entry.GuidValue

Description	Specifies the GUID value for a PPI.
Required	YES
Data Type	Element – RegistryFormatGuid
Data Constraints	The Registry Format GUID value.
Examples	`<GuidValue> AF0DDA2E-EA83-480b-B2CE-FC0BB2F894C2 </GuidValue>`

5.9.7 PpiDeclarations.Entry.HelpText

Table 90. PpiDeclarations.Entry.HelpText

Description	A complete description of a PPI. This must include any functions, defines, and data structures associated with this PPI. It must also describe any use restrictions based on Module Type, CPU Architecture, and/or Feature Flags.
Required	NO
Data Type	Element – xs:string
Data Constraints	A string that contains one or more lines of text.
Examples	`<HelpText>The PCI CFG PPI provides services to access PCI Configuration Headers from a PEIM. This PPI is available to all CPU types, but is restricted for use by PEIM module types.</HelpText>`

5.9.8 PpiDeclarations.Entry.HelpText:Lang

Table 91. PpiDeclarations.Entry.HelpText:Lang

Description	The language code uses an RFC 1766 language code identifier
Required	NO
Data Type	Attribute – xs:language
Data Constraints	If present, the language code must use an RFC 1766 language code identifier. If not specified, the default is “en-us.’
Examples	`Lang=”en-us”`

5.10 PackageSurfaceArea.PcdDeclarations

The following is the description of the `PackageSurfaceArea.PcdDeclarations` instance:

```
<PcdDeclarations>
  <PcdEntry
    SupArchList=" ArchListType " {0,1}
    SupModList=" ModuleListType " {0,1} >
      <TokenSpaceGuidCname> xs:NCName </TokenSpaceGuidCname> {1}
      <Token> HexNumber </Token> {1}
      <CName> xs:NCName </CName> {1}
      <DatumType> PcdDatumTypes </DatumType> {1}
      <ValidUsage> PcdItemListType </ValidUsage> {1}
      <DefaultValue> xs:normalizedString </DefaultValue> {1}
      <MaxDatumSize> HexNumber </MaxDatumSize> {0,1}
      <HelpText
         Lang=" xs:language " {0,1} >
           xs:string
      </HelpText> {0,}
      <PcdError> ... </PcdError> {0,}
    </PcdEntry> {1,}
</PcdDeclarations>
```

Table 92. PackageSurfaceArea.PcdDeclarations

Description	Specifies a list of Pcd entries that are defined by this Package. One and Only One package can specify these entries.
Required	NO
Data Type	Element – Complex
Data Constraints	N/A
Examples	N/A

5.10.1 PcdDeclarations.PcdEntry

Table 93. PcdDeclarations.PcdEntry

Description	Specifies a single of Pcd entry that are defined by this Package. One and Only One package can specify this entry.
Required	YES
Data Type	Element – Complex
Data Constraints	N/A
Examples	N/A

5.10.2 PcdDeclarations.PcdEntry:SupArchList

Table 94. PcdDeclarations.PcdEntry:SupArchList

Description	Used to restrict the set of CPU Architectures that are allowed to use this PCD Entry. If this attribute is not specified, then this PCD Entry may be used with any CPU Architecture. If this attribute is specified, then only those modules that support a subset of the CPU architectures specified by this element may use this PCD Entry.
Required	NO
Data Type	Attribute – ArchListType
Data Constraints	If specified, must contain one or more of the supported CPU architectures separated by spaces. The supported CPU architectures include **IA32**, **X64**, **IPF** and **EBC**
Examples	**SupArchList="IA32"** **SupArchList="IA32 X64"** **SupArchList="EBC IPF X64"**

5.10.3 PcdDeclarations.PcdEntry:SupModList

Table 95. PcdDeclarations.PcdEntry:SupModList

Description	Used to restrict the set of module types that are allowed to use this PCD Entry. If this attribute is not specified, then this PCD Entry may be used with all module types. If this attribute is specified, then only those modules that have a module type that is a member of the set of module types specified by this element may use this PCD Entry.
Required	NO
Data Type	Attribute – ModuleListType
Data Constraints	If specified, must contain one or more of the supported module types separated by spaces. The supported module types include **BASE**, **SEC**, **PEI_CORE**, **PEIM**, **DXE_CORE**, **DXE_DRIVER**, **DXE_RUNTIME_DRIVER**, **DXE_SAL_DRIVER**, **DXE_SMM_DRIVER**, **TOOL**, **UEFI_DRIVER**, **UEFI_RUNTIME_DRIVER**, **UEFI_APPLICATION**, **USER_DEFINED**.
Examples	**SupModList="BASE"** **SupModList="PEIM DXE_DRIVER"** **SupModList="UEFI_DRIVER UEFI_APPLICATION DXE_DRIVER"**

5.10.4 PcdDeclarations.PcdEntry.TokenSpaceGuidCname

Table 96. PcdDeclarations.PcdEntry.TokenSpaceGuidCname

Description	Specifies the C name of the Token Space GUID of which this PCD Entry is a member. This C name should also be listed in the GUIDs section (specified above) where the C name is assigned to a GUID value in C data structure format.
Required	YES
Data Type	Element – xs:NCName
Data Constraints	A valid C name

Examples	`<TokenSpaceGuidCName> gPcdBaseAddress</ TokenSpaceGuidCName>`

5.10.5 PcdDeclarations.PcdEntry.Token

Table 97. PcdDeclarations.PcdEntry.Token

Description	Specified the 32-bit token value for the PCD Entry
Required	YES
Data Type	Element – HexNumber
Data Constraints	Each PCD Entry in a Token Space must have a unique Token value. This element must contain one to eight hexadecimal characters.
Examples	**<Token>0x00010001</Token>** **<Token>0xA</Token>**

5.10.6 PcdDeclarations.PcdEntry.CName

Table 98. PcdDeclarations.PcdEntry.CName

Description	Specifies the symbol name for a PCD Entry used in C code. This element is also used by the UI Tools to display the name of the PCD Entry.
Required	YES
Data Type	Element – xs:NCName
Data Constraints	A string that starts with either an underscore or a letter, followed by any number of letters, digits or underscores.
Examples	**<CName> PcdDebugPropertyMask </CName>**

5.10.7 PcdDeclarations.PcdEntry.DatumType

Table 99. PcdDeclarations.PcdEntry.DatumType

Description	Specifies the datum type of this PCD Entry.
Required	YES
Data Type	Element – PcdDataType
Data Constraints	A string that contains the data type of the PCD Entry. PCD data types are restricted to the following set:**UINT8**, **UINT16**, **UINT32**, **UINT64**, **VOID***, **BOOLEAN**.
Examples	**<DatumType> UINT8 </DatumType>** **<DatumType> VOID* </DatumType>** **<DatumType> BOOLEAN </DatumType>**

5.10.8 PcdDeclarations.PcdEntry.ValidUsage

Table 100. PcdDeclarations.PcdEntry.ValidUsage

Description	Specifies the set of valid usages for this PCD Entry.
Required	YES

Data Type	Element – PcdItemListType
Data Constraints	A string that contains one or more PCD Entry item types separated by spaces. The PCD Entry item types are restricted to **FeaturePcd**, **FixedPcd**, **PatchPcd**, **Pcd** and/or **PcdEx**.
Examples	**<ValidUsage>** *FeaturePcd* **</ValidUsage>** **<ValidUsage>** *FixedPcd PatchPcd***</ValidUsage>** **<ValidUsage>** *FixedPcd PatchPcd Pcd PcdEx* **</ValidUsage>**

Pcd Item Types

FeaturePcd

The is a boolean PCD, which can cause different execution paths within a module, as well as to include or exclude different elements of a module during a build.

FixedPcd

This specifies that the value for this PCD is fixed at build time, and that it can only be modified at build time.

PatchPcd

This specifies that the value for this PCD is set during the build, but that binary information about this PCD must be provided, so that the value can be changed prior to adding it to a firmware volume, modified within a firmware volume or modified within a final flash image prior to installing the image into a firmware device.

Pcd

This usage is overloaded, with two different meanings. Prior to a build, the platform integrator can select whether the PCD will be fixed at build time, patchable, or truly dynamic. A truly dynamic instance requires the addition of a PEI and/or DXE pcd driver within a platform to maintain a volatile database of values that can set or retrieved. The data store for these entries is platform specific.

PcdEx

This specifies that the PCD is always a truly dynamic instance. Truly dynamic PCD entries require the addition of a PEI and/or DXE pcd driver within a platform to maintain a volatile database of values that can set or retrieved. The data store for these entries is platform specific.

5.10.9 PcdDeclarations.PcdEntry.DefaultValue

Table 101. PcdDeclarations.PcdEntry.DefaultValue

Description	Specifies the default value for the PCD Entry. This default value is only used to initialize the contents of a platform description or flash definition file when a module is added to a platform.
Required	YES
Data Type	Element – xs:normalizedString
Data Constraints	This element must support strings used to encode all the supported datum types, which includes 8, 16, 32, and 64 bit unsigned integers, Booleans, and buffers. The buffer types supported are ASCII strings, Unicode strings, and arrays of byte values.

Examples	`<DefaultValue>0x1f</DefaultValue>`
	`<DefaultValue>0x80000000</DefaultValue>`
	`<DefaultValue>234</DefaultValue>`
	`<DefaultValue>PlatformName</DefaultValue>`
	`<DefaultValue>LPlatformName</DefaultValue>`
	`<DefaultValue>0x01 0x02 0x03 0x45</DefaultValue>`

5.10.10 PcdDeclarations.PcdEntry.MaxDatumSize

Table 102. PcdDeclarations.PcdEntry.MaxDatumSize

Description	Specifies the Maximum Data Size for the PCD Entry. This value is used for VOID* DatumType, and should be the Maximum length required for the PCD's typical usage. Platform description files may over-ride this value – if and only if they specify a larger data size.
Required	YES – Only for VOID* DatumType PCDs
Data Type	Element – HexNumber
Data Constraints	This element is used for data structures to eliminate buffer-overruns.
Examples	`<MaxDatumSize>0x1f</MaxDatumSize>` `<MaxDatumSize> 0x20000 </MaxDatumSize>`

5.10.11 PcdDeclarations.PcdEntry.HelpText

Table 103. PcdDeclarations.PcdEntry.HelpText

Description	A complete description of a PCD Entry including the intended use models of the PCD Entry, its valid usage, its data type, and any value assignment constraints. It must also describe any use restrictions based on Module Type, CPU Architecture, and/or Feature Flags.
Required	YES
Data Type	Element – xs:string
Data Constraints	A string that contains one or more lines of text.

Examples	**<HelpText> The PcdDebugPropertyMask is an 8-bit mask of debug properties that may be set to a unique value for each module in the platform. This PCD Entry is available to all CPU and Module types and may be configured as FIXED_AT_BUILD or BINARY_PATCHABLE and supports the following bit settings: #define DEBUG_PROPERTY_DEBUG_ASSERT_ENABLED 0x01 #define DEBUG_PROPERTY_DEBUG_PRINT_ENABLED 0x02 #define DEBUG_PROPERTY_DEBUG_CODE_ENABLED 0x04 #define DEBUG_PROPERTY_CLEAR_MEMORY_ENABLED 0x08 </HelpText>**

5.10.12 PcdDeclarations.PcdEntry.HelpText:Lang

Table 104. PcdDeclarations.PcdEntry.HelpText:Lang

Description	The language code uses an RFC 1766 language code identifier
Required	NO
Data Type	Attribute – xs:language
Data Constraints	If present, the language code must use an RFC 1766 language code identifier. If not specified, the default is “en-us.’
Examples	**Lang=”en-us”** **Lang=”fra”**

5.10.13 PcdDeclarations.PcdEntry.PcdError

```
<PcdError>
  Choice (One and only one of the following must be selected)
   <ValidValueList
    Lang=" xs:language " {0,1} >
      xs:normalizedString
   </ValidValueList>
   <ValidValueRange> xs:normalizedString </ValidValueRange>
   <Expression> xs:normalizedString </Expression>
  End Choice
  <ErrorNumber> HexNumber </ErrorNumber> {1}
  <ErrorMessage
    Lang=" xs:language " {0,1} >
      xs:string
  </ErrorMessage> {1,}
</PcdError>
```

Table 105. PcdDeclarations.PcdEntry.PcdError

Description	Valid Error messages implemented in this module for the PCD Entry. Only One Error Number per PcdError, (multiple ErrorMessage entries are permitted) and multiple PcdError elements are permitted.
Required	NO
Data Type	Element – Complex
Data Constraints	N/A
Examples	N/A

5.10.14 PcdDeclarations.PcdEntry.PcdError.ValidValueList

Table 106. PcdDeclarations.PcdEntry.PcdError.ValidValueList

Description	A list of valid values
Required	NO

Data Type	Element – xs:normalizedString
Data Constraints	Space Separated list of values. Restricted to the data type of this PCD. Use of the L"string" is permitted to specify a UTF-8 text string that will be converted to UNICODE (UCS2) format in the code. Words that are not quoted are considered to be enumerations.

Examples	`<ValidValueList> 0xb0000000 0xb0000001 </ ValidValueList>`
	`<ValidValueList> "Blue" "Red" "White" </ ValidValueList>`
	`<ValidValueList> orange peach apple pear </ ValidValueList>`

5.10.15 PcdDeclarations.PcdEntry.PcdError.ValidValueList:Lang

Table 107. PcdDeclarations.PcdEntry.PcdError.ValidValueList:Lang

Description	The language code uses an RFC 1766 language code identifier
Required	NO
Data Type	Attribute – xs:language
Data Constraints	If present, the language code must use an RFC 1766 language code identifier. If not specified, the default is “en-us.’
Examples	**Lang=”en-us”** **Lang=”fra”**

5.10.16 PcdDeclarations.PcdEntry.PcdError.ValidValueRange

Table 108. PcdDeclarations.PcdEntry.PcdError.ValidValueRange

Description	The valid range of values using C style nomenclature.
Required	NO
Data Type	Element – xs:normalizedString
Data Constraints	This is required to be an in-fix logical expression, evaluated left to right, for a range of values, using Relational, Equality and Logical Operators (NOT, OR, AND, LT, LE, EQ, GT, GE and XOR) that must evaluate to either True or False. The forms PcdCName and/or PcdTokenSpaceGuidCName.PcdCName are permitted; if only the PcdCName is specified, then the PcdTokenSpaceGuidCName is the same as this PCD's TokenSpaceGuidCName. Parentheses are recommended for clarity. All other values must be numeric.
Examples	**<ValidValueRange> (0xb0000000 LE PcdCNameGoesHere) AND (PcdCNameGoesHere GT 0x00010000) </ ValidValueRange>** **<ValidValueRange> PcdCNameGoesHere LT 0x10001FFF AND (PcdCNameGoesHere NOT EQ 0x8000000) </ ValidValueRange>**

5.10.17 PcdDeclarations.PcdEntry.PcdError.Expression

Table 109. PcdDeclarations.PcdEntry.PcdError.Expression

Description	An in-fix logical expression, evaluated left to right, using Relational, Equality and Logical Operators (NOT, AND, OR, GT, GE, EQ, LE, LT and XOR) and parenthesis are recommended. The forms PcdCName and/or PcdTokenSpaceGuidCName.PcdCName are permitted; if only the PcdCName is specified, then the PcdTokenSpaceGuidCName is the same as this PCD's TokenSpaceGuidCName.
Required	NO

Data Type	Element – xs:normalizedString
Data Constraints	The expression must evaluate to true or false.
Examples	`<Expression> (NOT gPcdRecoveryMode) AND gPcdS3Resume </Expression>`

5.10.18 PcdDeclarations.PcdEntry.PcdError.ErrorNumber

Table 110. PcdDeclarations.PcdEntry.PcdError.ErrorNumber

Description	Valid error number per specifications.
Required	YES
Data Type	Element – HexNumber
Data Constraints	A hexadecimal value for the error message as defined by specifications.
Examples	`<ErrorNumber>0xb0000000</ErrorNumber>` `<ErrorNumber>0xb0000001</ErrorNumber>`

5.10.19 PcdDeclarations.PcdEntry.PcdError.ErrorMessage

Table 111. PcdDeclarations.PcdEntry.PcdError.ErrorMessage

Description	An error message for conditional failures of the choices above.
Required	YES
Data Type	Element – xs:string
Data Constraints	A string that contains one or more lines of text.
Examples	`<ErrorMessage>  The value specified exceeds the maximum value allowed. </ErrorMessage>`

5.10.20 PcdDeclarations.PcdEntry.PcdError.ErrorMessage:Lang

Table 112. PcdDeclarations.PcdEntry.PcdError.ErrorMessage:Lang

Description	The language code uses an RFC 1766 language code identifier
Required	NO
Data Type	Attribute – xs:language
Data Constraints	If present, the language code must use an RFC 1766 language code identifier. If not specified, the default is “en-us.’
Examples	`Lang=”en-us”` `Lang=”fra”`

5.11 PackageSurfaceArea.PcdRelationshipChecks

The following is the description of the **`PackageSurfaceArea.RelationshipChecks`** instance:

```
<PcdRelationshipChecks>
  <PcdCheck> xs:normalizedString </PcdCheck> {1,}
<PcdRelationshipChecks>
```

Table 113. PackageSurfaceArea.PcdRelationshipChecks

Description	This element is used to list any dependency or relationship expressions for PCDs at a global level.
Required	NO
Data Type	Element – Complex
Data Constraints	N/A
Examples	N/A

5.11.1 PcdRelationshipChecks.PcdCheck

Table 114. PcdRelationshipChecks.PcdCheck

Description	If one or more of this package's PCD settings may be dependent on a different PCD or have a specific relationship, then the relationship can be specified here. Each PCD is named by TokenSpaceGuidCname.PcdCname.
Required	YES
Data Type	Element – xs:normalizeString
Data Constraints	An in-fix logical expression, evaluated left to right, PCDs that are interdependent or have a limiting relationship, using Relational, Equality and Logical Operators (NOT, OR, AND, LT, LE, EQ, GT, GE and XOR) that must evaluate to either True or False. Only the C name for this PCD is permitted in the ValidValueRange expression. Parentheses are recommended for clarity. All other values must be numeric.
Examples	`<PcdCheck> gIoSerialTokenSpace.myComPort1Int XOR gIoSerialTokenSpace.myComPort2Int </PcdCheck> <PcdCheck> gIoTokenSpace.mPciBaseAddress LT gVideoTokenSpace.mNoSuchVideoDriverBaseAddress </ PcdCheck>`

5.12 PackageSurfaceArea.MiscellaneousFiles

The following is the description of the **`PackageSurfaceArea.MiscellaneousFiles`** instance:

```
<MiscellaneousFiles>
  <Copyright> xs:string </Copyright> {0,1}
  <License> xs:string </License> {0,1}
  <Abstract> xs:string </Abstract> {0,1}
  <Description> xs:string </Description> {0,}
  <Filename
    Executable=" xs:boolean " {0,1} >
      xs:anyURI
  </Filename> {1,}
</MiscellaneousFiles>
```

Table 115. PackageSurfaceArea.MiscellaneousFiles

Description	This element contains any miscellaneous files that are not part of code distributed with this package.
Required	NO
Data Type	Element – Complex
Data Constraints	Free-form text or any well formed XML element(s.)
Examples	N/A

5.12.1 MiscellaneousFiles.Copyright

Table 116. MiscellaneousFiles.Copyright

Description	The copyright for this section if it is different than the copyright specified in the package header. The copyright is generated by the creator of the tool. If a derivative work is generated from an existing section, then the existing copyright must be maintained, and additional copyrights may be appended to the end of this element.
Required	NO
Data Type	Element – xs:string
Data Constraints	A set of one or more copyright statements on one or more lines of text.
Examples	`<Copyright> Copyright (c) 2008, Nosuch Corporation. All rights reserved. </Copyright>`

5.12.2 MiscellaneousFiles.License

Table 117. MiscellaneousFiles.License

Description	A license that describes any restrictions on the use of this tool if it is different than the license specified in the package header. If a derivative work is allowed by the original license and a derivative work is generated from an existing package, then the existing license must be maintained, and additional licenses may be appended to the end of this element.
Required	NO
Data Type	Element – xs:string

Data Constraints	A quoted string that contains one or more lines of text.
Examples	**<License>** **This program and the accompanying materials are licensed and made available under the terms and conditions of the BSD License which accompanies this distribution. The full text of the license may be found at** **http://opensource.org/licenses/bsd-license.php** **THE PROGRAM IS DISTRIBUTED UNDER THE BSD LICENSE ON AN 'AS IS' BASIS, WITHOUT WARRANTIES OR REPRESENTATIONS OF ANY KIND, EITHER EXPRESS OR IMPLIED. </License>**

5.12.3 MiscellaneousFiles.Abstract

Table 118. MiscellaneousFiles.Abstract

Description	A short string for content in this section.
Required	NO
Data Type	Element – xs:string
Data Constraints	A string that starts with a letter followed by any combination of letters, digits, underscores, and periods. Whitespace characters are allowed.
Examples	`<Abstract>Special Build Rules 1 for processing file type foo</Abstract>`

5.12.4 MiscellaneousFiles.Description

Table 119. MiscellaneousFiles.Description

Description	Any additional detailed information that might be appropriate for content in this section.
Required	NO
Data Type	Element – xs:string
Data Constraints	A string that starts with a letter followed by any combination of letters, digits, underscores, and periods. Whitespace characters are allowed.
Examples	`<Description> Detailed instructions for using this package. </Description>`

5.12.5 MiscellaneousFiles.Filename

Table 120. MiscellaneousFiles.Filename

Description	A file that is included in the distribution package that does not pertain to any of the previously defined sections.
Required	YES
Data Type	Element – xs:anyURI
Data Constraints	The PackagePath relative path and filename of the file in the Distribution Content File.
Examples	`<File>FooBarMfgBuild/ReadMe.txt</File>` `<File> FooBarSpecification.pdf </File>`

5.12.6 MiscellaneousFiles.Filename:Executable

Table 121. MiscellaneousFiles.Filename:Executable

Description	This flag is used during installation to ensure that the file is executable.
Required	NO

Data Type	Attribute – xs:boolean
Data Constraints	"**true**" or "**false**" (default)
Examples	**Executable="true"**

5.13 PackageSurfaceArea.UserExtensions

The following is the description of the **PackageSurfaceArea.UserExtensions** instance:

```
<UserExtensions
  UserId=" xs:NCName " {1}
  Identifier=" xs:string " {1}
  AnyAttribute {0,} >
    Any Text or XML Format
</UserExtensions>
```

Table 122. PackageSurfaceArea.UserExtensions

Description	This section is used for any processing instructions that may be custom to the content provided by the distribution that are common to package. Processing of this section is specified as “lax”.
Required	NO
Data Type	Element – Complex
Data Constraints	None
Examples	N/A

5.13.1 UserExtensions:UserId

Table 123. UserExtensions:UserId

Description	The normative reference name to identify the originator of this information.
Required	YES
Data Type	Attribute – xs:NCName
Data Constraints	A string that starts with a letter followed by any combination of letters, digits, underscores, and periods. No whitespace characters are allowed.
Examples	`UserId=”NoSuchCorp”`

5.13.2 UserExtensions:Identifier

Table 124. UserExtensions:Identifier

Description	This can be used to differentiate multiple sections with a grouping.
Required	NO
Data Type	Attribute– xs:string
Data Constraints	A string that starts with a number or letter followed by any combination of letters, digits, underscores, dashes and periods. Whitespace characters are allowed.

Examples	`Identifier=”Special Build Rules 1”`
	`Identifier="f6665cf5-8290-4b02-ba0c-5cb5a9542176"`
	`Identifier=”PRE_BUILD”`

6
Module Surface Area Description

A distribution surface area may contain zero or more Module Surface Area sections. Within the Module Surface Area only the attributes and the Header section are required. The remaining sections are optional.

The XML Instance Representation for **DistributionPackage.ModuleSurfaceArea** is as follows:

```
<ModuleSurfaceArea
  BinaryModule=" xs:boolean " {0,1} >
    <Header> ... </Header> {1}
    <ModuleProperties> ... </ModuleProperties> {0,1}
    <ClonedFrom> ... </ClonedFrom> {0,1}
    <LibraryClassDefinitions> ... </LibraryClassDefinitions>
{0,1}
    <SourceFiles> ... </SourceFiles> {0,1}
    <BinaryFiles> ... </BinaryFiles> {0,1}
    <PackageDependencies> ... </PackageDependencies> {0,1}
    <Guids> ... </Guids> {0,1}
    <Protocols> ... </Protocols> {0,1}
    <PPIs> ... </PPIs> {0,1}
    <Externs> ... </Externs> {0,1}
    <PcdCoded> ... </PcdCoded> {0,1}
    <PeiDepex> ... </PeiDepex> {0,1}
    <DxeDepex> ... </DxeDepex> {0,1}
    <SmmDepex> ... </SmmDepex> {0,1}
    <MiscellaneousFiles> ... </MiscellaneousFiles> {0,1}
    <UserExtensions> ... </UserExtensions> {0,}
</ModuleSurfaceArea>
```

Table 125. ModuleSurfaceArea

Description	Specifies contents of a Module that is provided by this distribution package.
Required	NO
Data Type	Element – Complex
Data Constraints	N/A
Examples	N/A

6.1 ModuleSurfaceArea:BinaryModule

Table 126. ModuleSurfaceArea:BinaryModule

Description	This attribute is used when the binaries are distributed for this module and no code generation from source files is required. If set, then the binary section should be used, and any files listed in the SourceFiles section do not have to be built.
Required	NO
Data Type	Attribute – xs:boolean
Data Constraints	For binary only modules, this value should be set to "**`true`**", Default (or non-specified) value is "**`false`**".
Examples	**`BinaryModule="true"`**

6.2 ModuleSurfaceArea.Header

The following is the description of the **`ModuleSurfaceArea.Header`** instance:

```
<Header>
  <Name
    BaseName=" xs:NCName " {1} >
      xs:normalizedString
  </Name> {1}
  <GUID
    Version=" xs:decimal " {1} >
      RegistryFormatGuid
  </GUID> {1}
  <Copyright> xs:string </Copyright> {0,1}
  <License> xs:string </License> {0,1}
  <Abstract> xs:normalizedString </Abstract> {0,1}
  <Description> xs:string </Description> {0,1}
</Header>
```

Table 127. ModuleSurfaceArea.Header

Description	This element contains the header information for Module Surface Area Description. This includes the name of the module, descriptions of the module, copyright and licensing information associated with the module, and the version of XML Schema to which this file conforms.
Required	YES
Data Type	Element – Complex
Data Constraints	N/A
Examples	N/A

6.2.1 Header.Name

Table 128. Header.Name

Description	The User Interface Name for the module. This name is only used by a user interface to display the name of the module.
Required	YES
Data Type	Element – xs:normalizedString
Data Constraints	String value of more than one character.
Examples	`<Name> USB 1.0 I/O Driver </Name>`

6.2.2 Header.Name:BaseName

Table 129. Header.Name:BaseName

Description	This is a single word BaseName that will be used to create a module meta-data file. This name should also be used to create output file names and directories.
Required	YES
Data Type	`Attribute – xs:NCName`

Data Constraints	It must be a word made up of alphanumeric characters, starting with an alpha character. Non-word characters: underscore "_" and dash "-" are permitted after the first character.
Examples	**BaseName="IntelIch7"** **BaseName="Foo_bar-1_ten"**

6.2.3 Header.GUID

Table 130. Header.GUID

Description	The 128-bit registry format GUID that is the unique name of a module. The GUID must change if a change is made that is not backward compatible.
Required	YES
Data Type	Element – RegistryFormatGuid
Data Constraints	A string that represents a GUID in registry format. If a module is backwards compatible with a previous release of the same module, then **ModuleSurfaceArea.Header.GUID** element must not be changed, and only the **ModuleSurfaceArea.Header.GUID:Version** element should be increased. If a module is not backward compatible with a previous release, then a new **ModuleSurfaceArea.Header.GUID** must be generated.
Examples	**<GUID Version="0.3"> AF0DDA2E-EA83-480b-B2CE-FC0BB2F894C2 </GUID>**

6.2.4 Header.GUID:Version

Table 131. Header.GUID:Version

Description	The version of this Module.
Required	YES
Data Type	Attribute – xs:decimal
Data Constraints	A string that contains a positive decimal number with a dot separator.
Examples	**Version="1.00"** **Version="1.02"** **Version="3.27"**

6.2.5 Header.Copyright

Table 132. Header.Copyright

Description	This field is require only if the copyright of this module is different from the copyright of the container package that provides this module. The copyright for this module that is generated by the creator of a module. If a derivative work is generated from an existing module, then the existing copyright must be maintained, and additional copyrights may be appended to the end of this element.
Required	NO
Data Type	Element – xs:string
Data Constraints	A set of one or more copyright statements on one or more lines of text.

Examples	**<Copyright>** **Copyright (c) 2006, Nosuch Corporation. All rights reserved.** **</Copyright>**

6.2.6 Header.License

Table 133. Header.License

Description	This field is only required if the module license is different from the container package license. A license that describes any restrictions on the use of this module. If a derivative work is allowed by the original license and a derivative work is generated from an existing module, then the existing license must be maintained, and additional licenses may be appended to the end of this element. Alternative, this may point to a file which is located in the same directory a the module description file.
Required	NO
Data Type	Element – xs:string
Data Constraints	A string that contains one or more lines of text.

Examples	`<License>`
	`This program and the accompanying materials are licensed and made available under the terms and conditions of the BSD License which accompanies this distribution. The full text of the license may be found at:`
	`http://opensource.org/licenses/bsd-license.php`
	`THE PROGRAM IS DISTRIBUTED UNDER THE BSD LICENSE ON AN AS IS BASIS, WITHOUT WARRANTIES OR REPRESENTATIONS OF ANY KIND, EITHER EXPRESS OR IMPLIED.`
	`</License>`
	`<License> License.txt </License>`

6.2.7 Header.Abstract

Table 134. Header.Abstract

Description	A brief text description of the module. This description must include the release name of the module, the version of the module, and a description of the module contents and/or features.
Required	NO
Data Type	Element – xs:normalizedString
Data Constraints	A string that contains two or more words.
Examples	`<Abstract> USB I/O Driver Version 1.0 </Abstract>`

6.2.8 Header.Description

Table 135. Header.Description

Description	A complete description of the module. This description must include the release name of the module, the version of the module, and a complete description of the module contents and/or features including a description of the updates since the previous module release.
Required	NO
Data Type	Element – xs:string
Data Constraints	A string that contains one or more lines of text.

Examples	`<Description>`
	`USB I/O Driver Version 1.0 that conforms to the USB 2.2 specification. The module handles all USB I/O functions.`
	`</Description>`

6.3 ModuleSurfaceArea.ModuleProperties

The following is the description of the **`ModuleSurfaceArea.ModuleProperties`** instance:

```
<ModuleProperties
  SupArchList=" ArchListType " {0,1}
  SupModList=" ModuleListType " {0,1} >
    <ModuleType> ModuleTypes </ModuleType> {1}
    <Path> xs:anyURI </Path> {1}
    <PcdIsDriver> PcdDriverTypes </PcdIsDriver> {0,1}
    <UefiSpecificationVersion> xs:decimal </
UefiSpecificationVersion> {0,1}
    <PiSpecificationVersion> xs:decimal </
PiSpecificationVersion> {0,1}
    <Specification
      Version=" xs:decimal " >
        xs:NCName
    </Specification> {0,}
    <BootMode> ... </BootMode> {0,}
    <Event> ... </Event> {0,}
    <HOB> ... </HOB> {0,}
</ModuleProperties>
```

Table 136. ModuleSurfaceArea.ModuleProperties

Description	List general information about a module, including the Supported Architectures, the version of the UEFI and PI specs a module may conform to, and non-GUIDed information, such as boot modes a module might set, timer events, or hand off blocks.
Required	NO
Data Type	Element – Complex
Data Constraints	N/A
Examples	N/A

6.3.1 ModuleProperties:SupArchList

Table 137. ModuleProperties:SupArchList

Description	Used to restrict the set of CPU Architectures that are allowed to use this module. If this attribute is not specified, then this module may be used with any CPU Architecture. If this attribute is specified, then only those platforms that support a subset of the CPU architectures specified by this element may use this module. The list of Architectures that this module was designed to support and that it has been tested on.
Required	NO
Data Type	Attribute – ArchListType
Data Constraints	If specified, must contain one or more of the supported CPU architectures separated by spaces. The supported CPU architectures include **IA32**, **X64**, **IPF** and **EBC**

Examples	**SupArchList=”IA32”** **SupArchList=”IA32 X64”** **SupArchList=”EBC IPF X64”**

6.3.2 ModuleProperties:SupModList

Table 138. ModuleProperties:SupModList

Description	Used by Libraries to restrict the set of modules that the code in this library supports.
Required	NO
Data Type	Attribute – ModuleListType
Data Constraints	If specified, must contain one or more of the supported module types separated by spaces. The supported module types include **BASE**, **SEC**, **PEI_CORE**, **PEIM**, **DXE_CORE**, **DXE_DRIVER**, **DXE_RUNTIME_DRIVER**, **DXE_SAL_DRIVER**, **DXE_SMM_DRIVER**, **UEFI_DRIVER**, **UEFI_RUNTIME_DRIVER**, **UEFI_APPLICATION**, **USER_DEFINED**.
Examples	**SupModList="BASE"** **SupModList="PEIM DXE_DRIVER"** **SupModList="UEFI_DRIVER UEFI_APPLICATION DXE_DRIVER"**

6.3.3 ModuleProperties.ModuleType

Table 139. ModuleProperties.ModuleType

Description	Define the Module type that this module is.
Required	YES
Data Type	Element – ModuleType
Data Constraints	If specified, must contain one and only one of the supported module types. The supported module types include **BASE**, **SEC**, **PEI_CORE**, **PEIM**, **DXE_CORE**, **DXE_DRIVER**, **DXE_RUNTIME_DRIVER**, **DXE_SAL_DRIVER**, **DXE_SMM_DRIVER**, **UEFI_DRIVER**, **UEFI_RUNTIME_DRIVER**, **UEFI_APPLICATION**, **USER_DEFINED**.
Examples	**<ModuleType> BASE </MODULE_TYPE>** **<ModuleType> PEIM </MODULE_TYPE>** **SupModList="UEFI_DRIVER"**

6.3.4 ModuleProperties.Path

Table 140. ModuleProperties.Path

Description	For stand-alone modules that are NOT part of any package, this is the path to the root of the module as listed in the Distribution Content File. For modules included in a package, this is the location, relative to the root of the package this module belongs to.
Required	YES
Data Type	Element – xs:anyURI
Data Constraints	A valid (Distribution Content File or Package) relative path to the root of the Module.

Examples	<Path>Libraries/BaseLib</Path> <Path> EfiShellBin </Path> <Path>Universal/Disk/DiskIo/Dxe</Path>

6.3.5 ModuleProperties.PcdIsDriver

Table 141. ModuleProperties.PcdIsDriver

Description	This element is only required for the PEIM that produces the PCD PPI or the DXE Driver that produces the PCD Protocol.
Required	NO
Data Type	Element – xs:NCName
Data Constraints	One of enumerated type: **PEI_PCD_DRIVER** or **DXE_PCD_DRIVER**
Examples	**<PcdIsDriver>PEI_PCD_DRIVER</PcdIsDriver>** **<PcdIsDriver>DXE_PCD_DRIVER</PcdIsDriver>**

6.3.6 ModuleProperties.UefiSpecificationVersion

Table 142. ModuleProperties.UefiSpecificationVersion

Description	The number should reflect the version of the UEFI specification that the module conforms to. This value should be provided.
Required	NO
Data Type	Element – xs:decimal
Data Constraints	A decimal value that must include the period character.
Examples	**<UefiSpecificationVersion> 2.1 </ UefiSpecificationVersion>**

6.3.7 ModuleProperties.PiSpecificationVersion

Table 143. ModuleProperties.PiSpecificationVersion

Description	The number should reflect the version of the PI specification that the module conforms to. This may be left blank.
Required	NO
Data Type	Element – xs:decimal
Data Constraints	A decimal value that must include the period character.
Examples	**<PiSpecificationVersion> 1.0 </ PiSpecificationVersion>**

6.3.8 ModuleProperties.Specification

Table 144. ModuleProperties.Specification

Description	One or more of these elements indicate that the module has been coded to conform to the specified Industry Standard Specifications.
Required	NO

Data Type	Element – xs:NCName
Data Constraints	NA
Examples	**<Specification Version=”1.0”> NoSuch_MACHINE_DOT_COM </Specification>**

6.3.9 ModuleProperties.Specification:Version

Table 145. ModuleProperties.Specification:Version

Description	The Industry Standard spec version of this Module. If a module is coded to support more than one version of a spec, such as support for USB 1.1 and USB 2.0 specifications, then two different Specification lines should be used.
Required	YES
Data Type	Attribute – xs:decimal
Data Constraints	A decimal value that must include the period character.

Examples	`Version="1.0"`
	`Version="1.1"`
	`Version="2.0"`

6.3.10 ModuleProperties.BootMode

```
<BootMode>
  Usage=" one of the enumerated values " {1}
  SupArchList=" ArchListType " {0,1}
  FeatureFlag=" xs:normalizedString " {0,1} >
   <SupportedBootModes> list of Boot Modes </SupportedBootModes>
{1}
    <HelpText
       Lang=" xs:language " {0,1} >
         xs:string
    </HelpText> {0,}
</BootMode> {1,}
```

Table 146. ModuleProperties.BootMode

Description	This is a list of boot modes supported or set by this module.
Required	NO
Data Type	Element – Complex
Data Constraints	N/A
Examples	N/A

6.3.11 ModuleProperties.BootMode:Usage

Table 147. ModuleProperties.BootMode:Usage

Description	This element specifies the GUID Usage.
Required	YES
Data Type	Attribute – UsageType

Data Constraints	One of the enumerated values: **CONSUMES**, **PRODUCES**, **SOMETIMES_CONSUMES** or **SOMETIMES_PRODUCES** **CONSUMES** means that the module supports the boot mode or that it may support the boot mode on some execution paths. **PRODUCES** means that the module will change the boot mode
Examples	**Usage="PRODUCES"** **Usage="CONSUMES"**

6.3.12 ModuleProperties.BootMode:SupArchList

Table 148. ModuleProperties.BootMode:SupArchList

Description	Used to restrict the set of CPU Architectures that are allowed to use this Boot Mode. If this attribute is not specified, then this Boot Mode may be used with any CPU Architecture. If this attribute is specified, then only those modules that support a subset of the CPU architectures specified by this element may use this BootMode.
Required	NO
Data Type	Attribute – ArchListType
Data Constraints	If specified, must contain one or more of the supported CPU architectures separated by spaces. The supported CPU architectures include **IA32**, **X64**, **IPF** and **EBC**
Examples	**SupArchList="IA32"** **SupArchList="IA32 X64"** **SupArchList="EBC IPF X64"**

6.3.13 ModuleProperties.BootMode:FeatureFlag

Table 149. ModuleProperties.BootMode:FeatureFlag

Description	Used to restrict the use of this Boot Mode. An in-fix logical expression, evaluated left to right, for a range of values, using Relational, Equality and Logical Operators (NOT, OR, AND, LT, LE, EQ, GT, GE and XOR) that must evaluate to either True or False. The forms PcdCName and/or PcdTokenSpaceGuidCName.PcdCName are permitted; if only the PcdCName is specified, then the PcdTokenSpaceGuidCName is the same as this PCD's TokenSpaceGuidCName. Parentheses are recommended for clarity. If this attribute is specified, then this Boot Mode may only be used if the Boolean expression evaluates to **true**.
Required	NO
Data Type	Attribute – xs:normalizedString
Data Constraints	PCD C names, true, false and/or Operators : NOT, AND, OR, XOR, EQ, LT, LE , GT and GE.
Examples	**FeatureFlag="true"**

6.3.14 ModuleProperties.BootMode.SupportedBootModes

Table 150. ModuleProperties.BootMode.SupportedBootMode

Description	Firmware execution paths are divided into two main categories, a normal boot and a recovery boot (in the case of potential corruption.) Within these two paths, different paths can be taken based on a given state of the firmware, or through feature settings. A **BootMode** GUID can be installed (**PRODUCES**) or located (**CONSUMES**) based on these states and feature settings. The majority of these types map to the PI specification Boot Mode Services. The boot modes listed with Recovery are to indicate that the GUID is only valid during a recovery boot.

Required	YES
Data Type	Element – BootModeList
Data Constraints	One or more (space separated) of the enumerated data types: **FULL**, **MINIMAL**,**NO_CHANGE**, **DIAGNOSTICS**, **DEFAULT**, **S2_RESUME**, **S3_RESUME**, **S4_RESUME**, **S5_RESUME**, **FLASH_UPDATE**, **RECOVERY_FULL**, **RECOVERY_MINIMAL**, **RECOVERY_NO_CHANGE**, **RECOVERY_DIAGNOSTICS**, **RECOVERY_DEFAULT**, **RECOVERY_S2_RESUME**, **RECOVERY_S3_RESUME**, **RECOVERY_S4_RESUME**, **RECOVERY_S5_RESUME**, **RECOVERY_FLASH_UPDATE**
Examples	**<SupportedBootModes> FULL RECOVERY_FULL </ SupportedBootModes>**

6.3.15 ModuleProperties.BootMode.HelpText

Table 151. ModuleProperties.BootMode.HelpText

Description	This element specifies the HelpText describing how the module either sets or supports specific boot modes.
Required	NO
Data Type	Element – xs:string
Data Constraints	A string that contains one or more lines of text.
Examples	`<HelpText> Initializes the Dxe Ipl PPI for all boot modes except S3 Resume.</HelpText>`

6.3.16 ModuleProperties.BootMode.HelpText:Lang

Table 152. ModuleProperties.BootMode.HelpText:Lang

Description	The language code uses an RFC 1766 language code identifier
Required	NO
Data Type	Attribute – xs:language
Data Constraints	If present, the language code must use an RFC 1766 language code identifier. If not specified, the default is “en-us.’
Examples	`Lang=”en-us”`

6.3.17 ModuleProperties.Event

```
<Event
  Usage=" UsageType " {1}
  EventType=" EventType " {1}
  SupArchList=" ArchListType " {0,1}
  FeatureFlag=" xs:normalizedString " {0,1} >
    <HelpText
      Lang=" xs:language " {0,1} >
        xs:string
    </HelpText> {0,}
</Event>
```

Table 153. ModuleSurfaceArea.ModuleProperties.Event

Description	This is a list of non-GUIDed Events produced or consumed by this module.
Required	NO
Data Type	Element – Complex, nillable=”true”
Data Constraints	N/A
Examples	N/A

6.3.18 ModuleProperties.Event:Usage

Table 154. ModuleProperties.Event:Usage

Description	This element specifies the non-GUIDed Event Usage.
Required	YES
Data Type	Attribute – UsageType
Data Constraints	One of the enumerated values: **CONSUMES**, **PRODUCES**, **SOMETIMES_CONSUMES** or **SOMETIMES_PRODUCES** **PRODUCES** means that a module will produce a GUID that does not fit into the defined PROTOCOL, PPI or other GUID types. **CONSUMES** means that a module may use this GUID that does not fit into the defined GUID types.
Examples	**Usage="PRODUCES"** **Usage="CONSUMES"**

6.3.19 ModuleProperties.Event:EventType

Table 155. ModuleProperties.Event:EventType

Description	This element specifies the type of Event.
Required	YES
Data Type	Attribute – UsageType
Data Constraints	One of the enumerated values: **EVENT_TYPE_PERIODIC_TIMER** or **EVENT_TYPE_RELATIVE_TIMER**
Examples	**EventType="EVENT_TYPE_RELATIVE_TIMER"**

6.3.20 ModuleProperties.Event:SupArchList

Table 156. ModuleProperties.Event:SupArchList

Description	Used to restrict the set of CPU Architectures that are allowed to use this GUID. If this attribute is not specified, then this GUID may be used with any CPU Architecture. If this attribute is specified, then only those modules that support a subset of the CPU architectures specified by this element may use this GUID.
Required	NO
Data Type	Attribute – ArchListType
Data Constraints	If specified, must contain one or more of the supported CPU architectures separated by spaces. The supported CPU architectures include **IA32**, **X64**, **IPF** and **EBC**
Examples	**SupArchList="IA32"** **SupArchList="IA32 X64"** **SupArchList="EBC IPF X64"**

6.3.21 ModuleProperties.Event:FeatureFlag

Table 157. ModuleProperties.Event:FeatureFlag

Description	Used to restrict the use of this GUID. This attribute is a Boolean expression containing PCD Feature Flag names and operators. It may be an in-fix logical expression, evaluated left to right, for a range of values, using Relational, Equality and Logical Operators (NOT, OR, AND, LT, LE, EQ, GT, GE and XOR) that must evaluate to either True or False. The forms PcdCName and/or PcdTokenSpaceGuidCName.PcdCName are permitted; if only the PcdCName is specified, then the PcdTokenSpaceGuidCName is the same as this PCD's TokenSpaceGuidCName. Parentheses are recommended for clarity. If this attribute is specified, then this GUID may only be used if the Boolean expression evaluates to **`true`**.
Required	NO
Data Type	Attribute – xs:normalizedString
Data Constraints	PCD C names, true, false and/or Operators : NOT, AND, OR, XOR, EQ, LT, LE, GT and GE.
Examples	**`FeatureFlag="true"`**

6.3.22 ModuleProperties.Event.HelpText

Table 158. ModuleProperties.Event.HelpText

Description	This element specifies the HelpText describing how the module sets or waits for a timer event.
Required	NO
Data Type	Element – xs:string
Data Constraints	A string that contains one or more lines of text.
Examples	**`<HelpText> If Password is NULL unlock the password state variable and set the event timer. If the Password is too big return an error. If the Password is valid, copy the Password and enable state variable and then arm the periodic timer. </ HelpText>`**

6.3.23 ModuleProperties.Event.HelpText:Lang

Table 159. ModuleProperties.Event.HelpText:Lang

Description	The language code uses an RFC 1766 language code identifier
Required	NO
Data Type	Attribute – xs:language
Data Constraints	If present, the language code must use an RFC 1766 language code identifier. If not specified, the default is “en-us.’

Examples	`Lang="en-us"`

6.3.24 ModuleProperties.HOB

```
<HOB
  Usage=" UsageType " {1}
  HobType=" HobType " {1}
  SupArchList=" ArchListType " {0,1}
  FeatureFlag=" xs:normalizedString " {0,1} >
    <HelpText
      Lang=" xs:language " {0,1} >
        xs:string
    </HelpText> {0,}
</HOB>
```

Table 160. ModuleSurfaceArea.ModuleProperties.HOB

Description	This is a list of non-GUIDed HOBs produced or consumed by this module.
Required	NO
Data Type	Element – Complex, nillable="true"
Data Constraints	N/A
Examples	N/A

6.3.25 ModuleProperties.HOB:Usage

Table 161. ModuleProperties.HOB:Usage

Description	This element specifies the HOB Usage.
Required	YES
Data Type	Attribute – UsageType
Data Constraints	One of the enumerated values: **CONSUMES**, **PRODUCES**, **SOMETIMES_CONSUMES** or **SOMETIMES_PRODUCES** **PRODUCES** means that a module will produce a HOB that does not fit into the defined PROTOCOL, PPI or other GUID types. **CONSUMES** means that a module may use this HOB that does not fit into the defined GUID types.
Examples	**Usage="PRODUCES"** **Usage="CONSUMES"**

6.3.26 ModuleProperties.HOB:HobType

Table 162. ModuleProperties.HOB:HobType

Description	This element specifies the type of HOB.
Required	YES
Data Type	Attribute – Enumerated list value

Data Constraints	One of the enumerated values: **PHIT**, **MEMORY_ALLOCATION**, **RESOURCE_DESCRIPTOR**, **FIRMWARE_VOLUME** or **LOAD_PEIM**
Examples	**HobType="PHIT"** **HobType="LOAD_PEIM"**

6.3.27 ModuleProperties.HOB:SupArchList

Table 163. ModuleProperties.HOB:SupArchList

Description	Used to restrict the set of CPU Architectures that are allowed to use this HOB. If this attribute is not specified, then this HOB may be used with any CPU Architecture. If this attribute is specified, then only those modules that support a subset of the CPU architectures specified by this element may use this GUID.
Required	NO
Data Type	Attribute – ArchListType
Data Constraints	If specified, must contain one or more of the supported CPU architectures separated by spaces. The supported CPU architectures include **IA32**, **X64**, **IPF** and **EBC**
Examples	**SupArchList="IA32"** **SupArchList="IA32 X64"**

6.3.28 ModuleProperties.HOB:FeatureFlag

Table 164. ModuleProperties.HOB:FeatureFlag

Description	Used to restrict the use of this HOB. This attribute is a Boolean expression containing PCD Feature Flag names and operators. This is required to be an in-fix logical expression, evaluated left to right, for a range of values, using Relational, Equality and Logical Operators (NOT, OR, AND, LT, LE, EQ, GT, GE and XOR) that must evaluate to either True or False. The forms PcdCName and/or PcdTokenSpaceGuidCName.PcdCName are permitted; if only the PcdCName is specified, then the PcdTokenSpaceGuidCName is the same as this PCD's TokenSpaceGuidCName. Parentheses are recommended for clarity. If this attribute is specified, then this HOB may only be used if the Boolean expression evaluates to **true**.
Required	NO
Data Type	Attribute – xs:normalizedString
Data Constraints	PCD C names, true, false and/or Operators : NOT, AND, OR, XOR, EQ, LT, LE, GT and GE.
Examples	**FeatureFlag="true"**

6.3.29 ModuleProperties.HOB.HelpText

Table 165. ModuleProperties.HOB.HelpText

Description	This element specifies the HelpText describing correct usage of the HOB.
Required	NO
Data Type	Element – xs:string
Data Constraints	A string that contains one or more lines of text.
Examples	**<HelpText> This Module uses the gEfiHobMemoryAllocStackGuid to build a new memory allocation HOB with old stack info with EfiConventionalMemory type to be reclaimed by DXE core. </HelpText>**

6.3.30 ModuleProperties.HOB.HelpText:Lang

Table 166. ModuleProperties.HOB.HelpText:Lang

Description	The language code uses an RFC 1766 language code identifier
Required	NO
Data Type	Attribute – xs:language
Data Constraints	If present, the language code must use an RFC 1766 language code identifier. If not specified, the default is "en-us.'
Examples	**Lang="en-us"**

6.4 ModuleSurfaceArea.ClonedFrom

The following is the description of the **`ModuleSurfaceArea.ClonedFrom`** instance:

```
<ClonedFrom>
   <GUID
     Version=" xs:decimal " {1} >
       RegistryFormatGuid
   </GUID> {1}
</ClonedFrom>
```

6.4.1 ModuleSurfaceArea.ClonedFrom

Table 167. ModuleSurfaceArea.ClonedFrom

Description	If the module sources were copied from an existing Module, then this Section can be used to track the lineage.
Required	NO
Data Type	Element – Complex
Data Constraints	N/A
Examples	N/A

6.4.2 ClonedFrom.GUID

Table 168. ClonedFrom.GUID

Description	If the module sources were copied from an existing Module, this is the GUID of the Module.
Required	YES
Data Type	Element – RegistryFormatGuid
Data Constraints	Registry format GUID of the Module this module was copied from.
Examples	`<GUID Version="2">27d67720-ea68-48ae-93da-a3a074c90e30</GUID>`

6.4.3 ClonedFrom.GuidValue:Version

Table 169. ClonedFrom.GuidValue:Version

Description	This is the version of the Module that this module was copied from.
Required	YES
Data Type	Attribute – xs:decimal
Data Constraints	Version number of the existing Module
Examples	`Version="1.0"`

6.5 ModuleSurfaceArea.LibraryClassDefinitions

The following is the description of the **`ModuleSurfaceArea.LibraryClassDefinitions`** instance:

```
<LibraryClassDefinitions>
  <LibraryClass
    Usage=" LibUsageType " {1}
    SupArchList=" ArchListType " {0,1}
    SupModList=" ModuleListType " {0,1}
    FeatureFlag=" xs:normalizedString " {0,1} >
      <Keyword> xs:NCName </Keyword> {1}
      <RecommendedInstance>
        <GUID
          Version=" xs:decimal " {0,1} >
            RegistryFormatGuid
        </GUID> {1}
      </RecommendedInstance> {0,1}
      <HelpText
         Lang=" xs:language " {0,1} >
          xs:string
      </HelpText> {0,}
  </LibraryClass> {1,}
</LibraryClassDefinitions>
```

Table 170. ModuleSurfaceArea.LibraryClassDefinitions

Description	A list the different Library Classes consumed by a driver, core and application module, or produced by a Library module.
Required	NO
Data Type	Element – Complex
Data Constraints	N/A
Examples	N/A

6.5.1 LibraryClassDefinitions.LibraryClass

Table 171. LibraryClassDefinitions.LibraryClass

Description	A Library Class
Required	YES
Data Type	Element – Complex
Data Constraints	N/A
Examples	N/A

6.5.2 LibraryClassDefinitions.LibraryClass:Usage

Table 172. LibraryClassDefinitions.LibraryClass:Usage

Description	The Usage attribute which defines whether a Library Instance might be needed by the module, or if this a Library Instance for the specified LibraryClass.

Required	YES
Data Type	Attribute – xs:NCName
Data Constraints	One of the enumerated values: **PRODUCES**, **CONSUMES** or **SOMETIMES_CONSUMES**
Examples	**Usage="PRODUCES"**

Parameters

CONSUMES

The module requires one and only one library instance for the Library Class specified in the Keyword sub-element.

PRODUCES

This module is a library instance of the Library Class specified in the Keyword sub-element.

SOMETIMES_CONSUMES

The module may need one and only one library instance for the Library Class specified in the Keyword sub-element under some conditions.

6.5.3 LibraryClassDefinitions.LibraryClass:SupArchList

Table 173. LibraryClassDefinitions.LibraryClass:SupArchList

Description	Used to restrict the set of CPU Architectures that are allowed to use this Library Class. If this attribute is not specified, then this Library Class may be used with any CPU Architecture. If this attribute is specified, then only those modules that support a subset of the CPU architectures specified by this element may use this Library Class.
Required	NO
Data Type	Attribute – ArchListType
Data Constraints	If specified, must contain one or more of the supported CPU architectures separated by spaces. The supported CPU architectures include **IA32**, **X64**, **IPF** and **EBC**
Examples	**SupArchList="IA32"** **SupArchList="IA32 X64"** **SupArchList="EBC IPF X64"**

6.5.4 LibraryClassDefinitions.LibraryClass:SupModList

Table 174. LibraryClassDefinitions.LibraryClass:SupModList

Description	Used to restrict the set of module types that are allowed to use this Library Class. If this attribute is not specified, then this Library Class may be used with all module types. If this attribute is specified, then only those modules that have a module type that is a member of the set of module types specified by this element may use this Library Class.
Required	NO
Data Type	Attribute – ModuleListType
Data Constraints	If specified, must contain one or more of the supported module types separated by spaces. The supported module types include **BASE**, **SEC**, **PEI_CORE**, **PEIM**, **DXE_CORE**, **DXE_DRIVER**, **DXE_RUNTIME_DRIVER**, **DXE_SAL_DRIVER**, **DXE_SMM_DRIVER**, **TOOL**, **UEFI_DRIVER**, **UEFI_RUNTIME_DRIVER**, **UEFI_APPLICATION**, **USER_DEFINED**.
Examples	**SupModList="BASE"** **SupModList="PEIM DXE_DRIVER"** **SupModList="UEFI_DRIVER UEFI_APPLICATION DXE_DRIVER"**

6.5.5 LibraryClassDefinitions.LibraryClass:FeatureFlag

Table 175. LibraryClassDefinitions.LibraryClass:FeatureFlag

Description	Used to restrict the use of this Library Class. This attribute is a Boolean expression containing PCD Feature Flag names and operators. This is required to be an in-fix logical expression, evaluated left to right, for a range of values, using Relational, Equality and Logical Operators (NOT, OR, AND, LT, LE, EQ, GT, GE and XOR) that must evaluate to either True or False. The forms PcdCName and/or PcdTokenSpaceGuidCName.PcdCName are permitted; if only the PcdCName is specified, then the PcdTokenSpaceGuidCName is the same as this PCD's TokenSpaceGuidCName. Parentheses are recommended for clarity. If this attribute is specified, then this Library Class may only be used if the Boolean expression evaluates to **`true`**.
Required	NO
Data Type	Attribute – xs:normalizedString
Data Constraints	PCD C names, true, false and/or Operators : NOT, AND, OR, XOR, EQ, LT, LE, GT and GE.
Examples	**`FeatureFlag="true"`**

Table 176. LibraryClassDefinitions.LibraryClass.Keyword

Description	Used by the UI tools to identify different instances of libraries that provide the library class.
Required	YES
Data Type	Element – xs:NCName
Data Constraints	A single word that starts with an alpha character followed by one or more alphanumeric characters. The underscore "_" character is the only non-alphanumeric character allowed.
Examples	**`<Keyword>BaseMemoryLib</Keyword>`** **`<Keyword>CacheMaintenanceLib</Keyword>`**

6.5.6 LibraryClassDefinitions.LibraryClass.RecommendedInstance

Table 177. LibraryClassDefinitions.LibraryClass.RecommendedInstance

Description	The recommended library instance as defined by the creator of the Library Class Header file.
Required	NO
Data Type	Element – Complex
Data Constraints	N/A
Examples	N/A

6.5.7 LibraryClassDefinitions.LibraryClass.RecommendedInstance.GUID

Table 178. LibraryClassDefinitions.LibraryClass.RecommendedInstance.GUID

Description	The GUID of the recommended library instance as defined by the creator of the Library Class Header file. This is a required element. If GUID is specified, and the Version attribute is not, then the algorithm to look up the recommended instance is to find the library instance highest version number and a matching GUID.
Required	YES
Data Type	Element – RegistryFormatGuid
Data Constraints	A string that represents a GUID in registry format. Not valid if no RecommendedInf attribute specified.
Examples	`<GUID>AF0DDA2E-EA83-480b-B2CE-FC0BB2F894C2</GUID>`

6.5.8 LibraryClassDefinitions.LibraryClass.RecommendedInstance.GUID:Version

Table 179. LibraryClassDefinitions.LibraryClass:RecommendedInstance.GUID:Version

Description	The version of the recommended library instance as defined by the creator of the Library Class Header file. If this attribute is not specified, then the recommended library instance is the highest version number with a matching GUID.
Required	NO
Data Type	Attribute – xs:decimal
Data Constraints	Unsigned decimal number.
Examples	`Version="1.0"` `Version="2.75"`

6.5.9 LibraryClassDefinitions.LibraryClass.HelpText

Table 180. LibraryClassDefinitions.LibraryClass.HelpText

Description	Used by the UI tools to provide information about this library class, it's uses and functionality.
Required	NO
Data Type	Element – xs:string
Data Constraints	A string that contains one or more lines of text.
Examples	`<HelpText> Base Memory Library that uses MMX registers for high performance. Optimized for use in DXE. </HelpText>`

6.5.10 LibraryClassDefinitions.LibraryClass.HelpText:Lang

Table 181. LibraryClassDefinitions.LibraryClass.HelpText:Lang

Description	The language code uses an RFC 1766 language code identifier

Required	NO
Data Type	Attribute – xs:language
Data Constraints	If present, the language code must use an RFC 1766 language code identifier. If not specified, the default is “en-us.’
Examples	`Lang=”en-us”`

6.6 ModuleSurfaceArea.SourceFiles

This section is required if and only if this Module is NOT a Binary Module. If this is a binary module, then this section is not permitted.

The following is the description of the **`ModuleSurfaceArea.SourceFiles`** instance:

```
<SourceFiles>
  <Filename
    Family=" FamilyType " {0,1}
    SupArchList=" ArchListType " {0,1}
    FeatureFlag=" xs:normalizedString " {0,1} >
      xs:anyURI
  </Filename> {1,}
</SourceFiles>
```

Table 182. ModuleSurfaceArea.SourceFiles

Description	This element contains a list of all text-based code (including Unicode text) files that are included with this module.
Required	NO
Data Type	Element – Complex
Data Constraints	N/A
Examples	N/A

6.6.1 SourceFiles.Filename

Table 183. SourceFiles.Filename

Description	Source files used by tools to create binary files.
Required	YES
Data Type	Element – xs:anyURI
Data Constraints	The path (relative to the Module “root” directory) and filename of the file (as specified in the Distribution Content File.)
Examples	**<Filename>SwitchStack.c</Filename>** **<Filename SupArchList=”IA32”>x86LowLevel.c</Filename>** **<Filename SupArchList=”IA32”>Ia32/Non-existing.c</Filename>**

6.6.2 SourceFiles.Filename:Family

Table 184. SourceFiles.Filename:Family

Description	The Family attribute is used to restrict usage to a given family of compilers, such as GCC or MSFT. Since not all code processing tools use the same syntax, especially for assembly, this field can be used to identify different syntax. If not specified, then any family can use this file.
Required	NO
Data Type	Attribute – FamilyType
Data Constraints	Enumerated Data type: **MSFT**, **GCC**

Examples	**Family="GCC"** **Family="MSFT"**

6.6.3 SourceFiles.Filename:SupArchList

Table 185. SourceFiles.Filename:SupArchList

Description	Used to restrict the set of CPU Architectures that are allowed to use this filename. If this attribute is not specified, then this filename may be used with any CPU Architecture. If this attribute is specified, then only those modules that support a subset of the CPU architectures specified by this element may use this filename.
Required	NO
Data Type	Attribute – ArchListType
Data Constraints	If specified, must contain one or more of the supported CPU architectures separated by spaces. The supported CPU architectures include **IA32**, **X64**, **IPF** and **EBC**
Examples	**SupArchList="IA32"** **SupArchList="IA32 X64"** **SupArchList="EBC IPF X64"**

6.6.4 SourceFiles.Filename:FeatureFlag

Table 186. SourceFiles.Filename:FeatureFlag

Description	Used to restrict the use of this filename. This attribute is a Boolean expression containing PCD Feature Flag names and operators. This is an in-fix logical expression, evaluated left to right, for a range of values, using Relational, Equality and Logical Operators (NOT, OR, AND, LT, LE, EQ, GT, GE and XOR) that must evaluate to either True or False. The forms PcdCName and/or PcdTokenSpaceGuidCName.PcdCName are permitted; if only the PcdCName is specified, then the PcdTokenSpaceGuidCName is the same as this PCD's TokenSpaceGuidCName. Parentheses are recommended for clarity. If this attribute is specified, then this filename may only be used if the Boolean expression evaluates to **true**.
Required	NO
Data Type	Attribute – xs:normalizedString
Data Constraints	PCD C names, true, false and/or Operators : NOT, AND, OR, XOR, EQ, LT, LE, GT and GE.
Examples	**FeatureFlag="true"**

6.7 ModuleSurfaceArea.BinaryFiles

If no Source Files are specified, then the files listed here will not be processed during a build. They are either PI FFS sections, UEFI images or Binary files.

The following is the description of the **ModuleSurfaceArea.BinaryFiles** instance:

```
<BinaryFiles>
  <BinaryFile>
    <Filename
      FileType=" FileType " {1}
      SupArchList=" ArchListType " {0,1}
      FeatureFlag=" FeatureFlagExpression " {0,1} >
        xs:anyURI
    </Filename> {1,}
    <AsBuilt> ... </AsBuilt> {0,}
  </BinaryFile> {1,}
</BinaryFiles>
```

Table 187. ModuleSurfaceArea.BinaryFiles

Description	This element contains a list of all binary files that are included with a module. If Source files have been specified, these files may be included in a module. Typically, this section is used for distribution of binary only code.
Required	NO
Data Type	Element – Complex
Data Constraints	N/A
Examples	N/A

6.7.1 BinaryFiles.BinaryFile

Table 188. BinaryFiles.BinaryFile

Description	User can enter any data that might help other developers using this source code.
Required	NO
Data Type	Element – Complex
Data Constraints	N/A
Examples	N/A

6.7.2 BinaryFiles.BinaryFile.Filename

Table 189. BinaryFiles.BinaryFile.Filename

Description	Module Relative path and filename for a binary file.
Required	YES
Data Type	Element – xs:anyURI
Data Constraints	The path (relative to the Module's "root" directory and filename of the file (as specified in the Distribution Content File.)

Examples	<Filename FileType="PE32">SwitchStack.efi</Filename> <Filename SupArchList="IA32" FileType="BIN" > IA32/LowLevel.bin </Filename> <Filename FileType="BIN" SupArchList="X64"> X64/LowLevel.bin </Filename>

6.7.3 BinaryFiles.BinaryFile.Filename:FileType

Table 190. BinaryFiles.BinaryFile.Filename:FileType

Description	Specify the type of binary file provided.
Required	YES
Data Type	Attribute – BinFileType
Data Constraints	One of the enumerated values: **GUID**, **FREEFORM**, **UEFI_IMAGE**, **PE32**, **PIC**, **PEI_DEPEX**, **DXE_DEPEX**, **TE**, **VER**, **UI**, **BIN**, **FV**
Examples	**FileType="PE32"** **FileType="UI"**

6.7.4 BinaryFiles.BinaryFile.Filename:SupArchList

Table 191. BinaryFiles.BinaryFile.Filename:SupArchList

Description	Used to restrict the set of CPU Architectures that are allowed to use this filename. If this attribute is not specified, then this filename may be used with any CPU Architecture. If this attribute is specified, then only those modules that support a subset of the CPU architectures specified by this element may use this filename.
Required	NO
Data Type	Attribute – ArchListType
Data Constraints	If specified, must contain one or more of the supported CPU architectures separated by spaces. The supported CPU architectures include **IA32**, **X64**, **IPF** and **EBC**
Examples	**SupArchList="IA32"** **SupArchList="IA32 X64"** **SupArchList="EBC IPF X64"**

6.7.5 BinaryFiles.BinaryFile.Filename:FeatureFlag

Table 192. BinaryFiles.BinaryFile.Filename:FeatureFlag

Description	Used to restrict the use of this filename. This attribute is a Boolean expression containing PCD Feature Flag names and operators. This is required to be an in-fix logical expression, evaluated left to right, for a range of values, using Relational, Equality and Logical Operators (NOT, OR, AND, LT, LE, EQ, GT, GE and XOR) that must evaluate to either True or False. The forms PcdCName and/or PcdTokenSpaceGuidCName.PcdCName are permitted; if only the PcdCName is specified, then the PcdTokenSpaceGuidCName is the same as this PCD's TokenSpaceGuidCName. Parentheses are recommended for clarity. If this attribute is specified, then this filename may only be used if the Boolean expression evaluates to **true**.
Required	NO
Data Type	Attribute – xs:normalizedString

Data Constraints	PCD C names, true, false and/or Operators : NOT, AND, OR, XOR, EQ, LT, LE, GT and GE.
Examples	`FeatureFlag="PcdSupportUpdateCapsuleRest"`

6.7.6 BinaryFiles.BinaryFile.AsBuilt

If this is a binary module distribution, this section contains information about how the module was coded, such as Compiler Tools, Flags, PCDs (only PatchPcd, Pcd and/or PcdEx) and Library Class Instances used to build the binary.

```
<AsBuilt>
  <PatchPcdValue> ... </PatchPcdValue> {0,}
  <PcdExValue> ... </PcdExValue> {0,}
  <LibraryInstances> ... </LibraryInstances> {0,1}
  <BuildFlags> Any attributes or content is permitted
<BuildFlags> {0,}
</AsBuilt> {0,}
```

Table 193. BinaryFiles.BinaryFile.AsBuilt

Description	This section covers the information required by developers receiving a binary distribution. 'If this is a binary module distribution, this section contains information about how the module was coded, such as Compiler Tools, Flags, PCDs (only PatchPcd, Pcd and/or PcdEx) and Library Class Instances used to build the binary.
Required	NO
Data Type	Element – Complex
Data Constraints	N/A
Examples	N/A

6.7.7 BinaryFiles.BinaryFile.AsBuilt.PatchPcdValue

```
<PatchPcdValue>
  <TokenSpaceGuidValue> RegistryFormatGuid </
TokenSpaceGuidValue> {1}
  <PcdCName> xs:NCName </PcdCName> {1}
  <Token> HexNumber </Token> {1}
  <DatumType> PcdDatumType </DatumType> {1}
  <MaxDatumSize> HexNumber </MaxDatumSize> {0,1}
  <Value> xs:normalizedString </Value> {1}
  <Offset> HexNumber </Offset> {1}
  <HelpText> ... </HelpText> {0,}
  <PcdError> ... </PcdError> {0,}
</PatchPcdValue>
```

Table 194. BinaryFiles.BinaryFile.AsBuilt.PatchPcdValue

Description	An optional section that can describes any patchable PCDs that may be pertinent to this binary module.
Required	NO
Data Type	Element – Complex
Data Constraints	N/A
Examples	N/A

6.7.8 BinaryFiles.BinaryFile.AsBuilt.PatchPcdValue.TokenSpaceGuidValue

Table 195. BinaryFiles.BinaryFile.AsBuilt.PatchPcdValue.TokenSpaceGuidValue

Description	The GUID value TokenSpaceGuid that defines this PCD.
Required	YES
Data Type	Element – RegistryFormatGuid
Data Constraints	Registry format GUID of the Token Space GUID.
Examples	`<TokenSpaceGuidValue> 914AEBE7-4635-459b-AA1C-11E219B03A10 </TokenSpaceGuidValue>`

6.7.9 BinaryFiles.BinaryFile.AsBuilt.PatchPcdValue.Token

Table 196. BinaryFiles.BinaryFile.AsBuilt.PatchPcdValue.Token

Description	The Token Number of this PCD.
Required	YES
Data Type	Element – HexNumber
Data Constraints	The hex token number used during the platform build. This element must contain one to eight hexadecimal characters.
Examples	`<Token>0x00000006</Token>`

6.7.10 BinaryFiles.BinaryFile.AsBuilt.PatchPcdValue.PcdCName

Table 197. BinaryFiles.BinaryFile.AsBuilt.PatchPcdValue.PcdCName

Description	The name of this PCD.
Required	YES
Data Type	Element – xs:NCName
Data Constraints	A string that starts with a letter followed by any combination of letters, digits, underscores, and periods. No whitespace characters are allowed.
Examples	`<PcdCName> PcdDebugPrintErrorLevel </PcdCName>`

6.7.11 BinaryFiles.BinaryFile.AsBuilt.PatchPcdValue.DatumType

Table 198. BinaryFiles.BinaryFile.AsBuilt.PatchPcdValue.DatumType

Description	The data type of this PCD.
Required	YES
Data Type	Element – xs:NCName
Data Constraints	One of the following enumerated data types: UINT8, UINT16, UINT32, UINT64, BOOLEAN, or VOID*
Examples	`<DatumType> VOID* </DatumType>`

6.7.12 BinaryFiles.BinaryFile.AsBuilt.PatchPcdValue.MaxDatumSize

Table 199. BinaryFiles.BinaryFile.AsBuilt.PatchPcdValue.MaxDatumSize

Description	The maximum data size allocated for this PCD; this is required for VOID* Pcd Datum Types.
Required	NO (Depending on the PCD's datum type)
Data Type	Element – HexNumber
Data Constraints	A hex value.
Examples	`<MaxDatumSize> 0x0000001F </MaxDatumSize>`

6.7.13 BinaryFiles.BinaryFile.AsBuilt.PatchPcdValue.Value

Table 200. BinaryFiles.BinaryFile.AsBuilt.PatchPcdValue.Value

Description	The value that was used at build time in the binary.
Required	YES
Data Type	Element – xs:normalizedString
Data Constraints	A string that starts with a letter followed by any combination of letters, digits, underscores, and periods. Whitespace characters are allowed.
Examples	`<Value>"en-US;fr-FR"<Value>`

6.7.14 BinaryFiles.BinaryFile.AsBuilt.PatchPcdValue.Offset

Table 201. BinaryFiles.BinaryFile.AsBuilt.PatchPcdValue.Offset

Description	The offset into the binary of this PCD.
Required	YES
Data Type	Element – HexNumber
Data Constraints	The hex address into the binary image of the PCD
Examples	`<Offset> 0x00000200 </Offset>`

6.7.15 BinaryFiles.BinaryFile.AsBuilt.PatchPcdValue.HelpText

Table 202. BinaryFiles.BinaryFile.AsBuilt.PatchPcdValue.HelpText

Description	A complete description of a how the PCD is implemented.
Required	NO
Data Type	Element – xs:string
Data Constraints	A string that contains one or more lines of text.

Examples	**<HelpText> The PcdDebugPropertyMask is an 8-bit mask of debug properties that may be set to a unique value for each module in the platform. This PCD Entry is available to all CPU and Module types and was configured as BINARY_PATCHABLE and the following bit settings: #define DEBUG_PROPERTY_DEBUG_ASSERT_ENABLED 0x01 #define DEBUG_PROPERTY_DEBUG_PRINT_ENABLED 0x02 </HelpText>**

6.7.16 BinaryFiles.BinaryFile.AsBuilt.PatchPcdValue.HelpText:Lang

Table 203. BinaryFiles.BinaryFile.AsBuilt.PatchPcdValue.HelpText:Lang

Description	The language code uses an RFC 1766 language code identifier
Required	NO
Data Type	Attribute – xs:language
Data Constraints	If present, the language code must use an RFC 1766 language code identifier. If not specified, the default is “en-us.’
Examples	**Lang=”en-us”** **Lang=”fra”**

6.7.17 BinaryFiles.BinaryFile.AsBuilt.PatchPcdValue.PcdError

```
<PcdError>
  <ErrorNumber> HexNumber </ErrorNumber> {1}
  <ErrorMessage
    Lang=" xs:language " {0,1} >
      xs:string
  </ErrorMessage> {0,}
</PcdError>
```

Table 204. BinaryFiles.BinaryFile.AsBuilt.PatchPcdValue.PcdError

Description	Valid Error messages implemented in this module for the PatchPcd entry. Only One Error Number per PcdError, (multiple ErrorMessage entries are permitted) and multiple PcdError elements are permitted.
Required	NO
Data Type	Element – Complex
Data Constraints	N/A
Examples	N/A

6.7.18 BinaryFiles.BinaryFile.AsBuilt.PatchPcdValue.PcdError.ErrorNumber

Table 205. BinaryFiles.BinaryFile.AsBuilt.PatchPcdValue.PcdError.ErrorNumber

Description	Valid error number per specifications.
Required	YES
Data Type	Element – HexNumber
Data Constraints	A hexadecimal value for the error message as defined by specifications.
Examples	**<ErrorNumber>0xb0000000</ErrorNumber>** **<ErrorNumber>0xb0000001</ErrorNumber>**

6.7.19 BinaryFiles.BinaryFile.AsBuilt.PatchPcdValue.PcdError.ErrorMessage

Table 206. BinaryFiles.BinaryFile.AsBuilt.PatchPcdValue.PcdError.ErrorMessage

Description	An error message for conditional failures of the choices above.
Required	NO
Data Type	Element – xs:string
Data Constraints	A string that contains one or more lines of text.
Examples	`<ErrorMessage> The value specified exceeds the maximum value allowed. </ErrorMessage>`

6.7.20 BinaryFiles.BinaryFile.AsBuilt.PatchPcdValue.PcdError.ErrorMessage:Lang

Table 207. BinaryFiles.BinaryFile.AsBuilt.PatchPcdValue.PcdError.ErrorMessage:Lang

Description	The language code uses an RFC 1766 language code identifier
Required	NO
Data Type	Attribute – xs:language
Data Constraints	If present, the language code must use an RFC 1766 language code identifier. If not specified, the default is “en-us.’
Examples	`Lang=”en-us”` `Lang=”fra”`

6.7.21 BinaryFiles.BinaryFile.AsBuilt.PcdExValue

```
<PcdExValue>
  <TokenSpaceGuidValue> RegistryFormatGuid </
TokenSpaceGuidValue> {1}
  <Token> HexNumber </Token> {1}
  <DatumType> PcdDatumType </DatumType> {1}
  <MaxDatumSize> HexNumber </MaxDatumSize> {0,1}
  <Value> xs:normalizedString </Value> {1}
  <HelpText> ... </HelpText> {0,}
  <PcdError> ... </PcdError> {0,}
</PcdExValue>
```

Table 208. BinaryFiles.BinaryFile.AsBuilt.PcdExValue

Description	An optional section that can describes any dynamic PCDs that may be pertinent to this binary module.
Required	NO
Data Type	Element – Complex
Data Constraints	N/A

Examples	N/A

6.7.22 BinaryFiles.BinaryFile.AsBuilt.PcdExValue.TokenSpaceGuidValue

Table 209. BinaryFiles.BinaryFile.AsBuilt.PcdExValue.TokenSpaceGuidValue

Description	The GUID value TokenSpaceGuid that defines this PCD.
Required	YES
Data Type	Element – RegistryFormatGuid
Data Constraints	Registry format GUID of the Token Space GUID.
Examples	`<TokenSpaceGuidValue> 914AEBE7-4635-459b-AA1C-11E219B03A10 </TokenSpaceGuidValue>`

6.7.23 BinaryFiles.BinaryFile.AsBuilt.PcdExValue.Token

Table 210. BinaryFiles.BinaryFile.AsBuilt.PcdExValue.Token

Description	The Token Number of this PCD.
Required	YES
Data Type	Element – HexNumber
Data Constraints	The hex token number used during the platform build. This element must contain one to eight hexadecimal characters.
Examples	`<Token>0x00000006</Token>`

6.7.24 BinaryFiles.BinaryFile.AsBuilt.PcdExValue.DatumType

Table 211. BinaryFiles.BinaryFile.AsBuilt.PcdExValue.DatumType

Description	The data type of this PCD.
Required	YES
Data Type	Element – xs:NCName
Data Constraints	One of the following enumerated data types: UINT8, UINT16, UINT32, UINT64, BOOLEAN, or VOID*
Examples	`<DatumType> VOID* </DatumType>`

6.7.25 BinaryFiles.BinaryFile.AsBuilt.PcdExValue.MaxDatumSize

Table 212. BinaryFiles.BinaryFile.AsBuilt.PcdExValue.MaxDatumSize

Description	The maximum data size allocated for this PCD; this is required for VOID* Pcd Datum Types.
Required	NO (Depending on the PCD's datum type)
Data Type	Element – HexNumber
Data Constraints	A hex value.

Examples	`<MaxDatumSize> 0x0000001F </MaxDatumSize>`

6.7.26 BinaryFiles.BinaryFile.AsBuilt.PcdExValue.Value

Table 213. BinaryFiles.BinaryFile.AsBuilt.PcdExValue.Value

Description	The value that was used at build time in the binary.
Required	YES
Data Type	Element – xs:normalizedString
Data Constraints	A string that starts with a letter followed by any combination of letters, digits, underscores, and periods. Whitespace characters are allowed.
Examples	**<Value>en-US;fr-FR<Value>**

6.7.27 BinaryFiles.BinaryFile.AsBuilt.PcdExValue.HelpText

Table 214. BinaryFiles.BinaryFile.AsBuilt.PcdExValue.HelpText

Description	A complete description of a how the PCD is implemented.
Required	NO
Data Type	Element – xs:string
Data Constraints	A string that contains one or more lines of text.
Examples	**<HelpText> Gets the base address of PCI Express. This internal functions retrieves PCI Express Base Address via a PCD entry PcdPciExpressBaseAddress and returns the base address of PCI Express. </ HelpText>**

6.7.28 BinaryFiles.BinaryFile.AsBuilt.PcdExValue.HelpText:Lang

Table 215. BinaryFiles.BinaryFile.AsBuilt.PcdExValue.HelpText:Lang

Description	The language code uses an RFC 1766 language code identifier
Required	NO
Data Type	Attribute – xs:language
Data Constraints	If present, the language code must use an RFC 1766 language code identifier. If not specified, the default is “en-us.’
Examples	**Lang=”en-us”** **Lang=”fra”**

6.7.29 BinaryFiles.BinaryFile.AsBuilt.PcdExValue.PcdError

```
<PcdError>
  <ErrorNumber> HexNumber </ErrorNumber> {1}
  <ErrorMessage
    Lang=" xs:language " {0,1} >
      xs:string
```

```
  </ErrorMessage> {0,}
</PcdError>
```

Table 216. BinaryFiles.BinaryFile.AsBuilt.PcdExValue.PcdError

Description	Valid Error messages implemented in this module for the dynamic PcdEx entry. Only One Error Number per PcdError elements (multiple ErrorMessage entries are permitted) and multiple PcdError elements are permitted.
Required	NO
Data Type	Element – Complex
Data Constraints	N/A
Examples	N/A

6.7.30 BinaryFiles.BinaryFile.AsBuilt.PcdExValue.PcdError.ErrorNumber

Table 217. BinaryFiles.BinaryFile.AsBuilt.PcdExValue.PcdError.ErrorNumber

Description	Valid error number per specifications.
Required	YES
Data Type	Element – HexNumber
Data Constraints	A hexadecimal value for the error message as defined by specifications.
Examples	`<ErrorNumber>0xb0000000</ErrorNumber>` `<ErrorNumber>0xb0000001</ErrorNumber>`

6.7.31 BinaryFiles.BinaryFile.AsBuilt.PcdExValue.PcdError.ErrorMessage

Table 218. BinaryFiles.BinaryFile.AsBuilt.PcdExValue.PcdError.ErrorMessage

Description	An error message for conditional failures of the choices above.
Required	NO
Data Type	Element – xs:string
Data Constraints	A string that contains one or more lines of text.
Examples	`<ErrorMessage>  The value specified exceeds the maximum value allowed. </ErrorMessage>`

6.7.32 BinaryFiles.BinaryFile.AsBuilt.PcdExValue.PcdError.ErrorMessage:Lang

Table 219. BinaryFiles.BinaryFile.AsBuilt.PcdExValue.PcdError.ErrorMessage:Lang

Description	The language code uses an RFC 1766 language code identifier

Required	NO
Data Type	Attribute – xs:language
Data Constraints	If present, the language code must use an RFC 1766 language code identifier. If not specified, the default is “en-us.’
Examples	`Lang=”en-us”` `Lang=”fra”`

6.7.33 BinaryFiles.BinaryFile.AsBuilt.LibraryInstances

```
<LibraryInstances>
  <GUID
    Version=" xs:decimal " {1} >
      RegistryFormatGuid
  </GUID>  {1,}
</LibraryInstances>
```

Table 220. BinaryFiles.BinaryFile.AsBuilt.LibraryInstances

Description	This section is used to define the Library Class Instances that were linked to the Module..
Required	NO
Data Type	Element – Complex
Data Constraints	N/A
Examples	N/A

6.7.34 BinaryFiles.BinaryFile.AsBuilt.LibraryInstances.GUID

Table 221. BinaryFiles.BinaryFile.AsBuilt.LibraryInstances:GUID

Description	The Registry Format GUID of the library Instance.
Required	YES
Data Type	Element – RegistryFormatGuid
Data Constraints	A string that represents a GUID in registry format.
Examples	`<GUID>fd44e603-002a-4b29-9f5f-529e815b6165</GUID>`

6.7.35 BinaryFiles.BinaryFile.AsBuilt.LibraryInstances.GUID:Version

Table 222. BinaryFiles.BinaryFile.AsBuilt.LibraryInstances:Version

Description	The Version Number of the Library Instance.
Required	NO
Data Type	Attribute – xs:decimal
Data Constraints	A decimal number
Examples	`Version=”1.0”`

6.7.36 BinaryFiles.BinaryFile.AsBuilt.BuildFlags

Table 223. BinaryFiles.BinaryFile.AsBuilt.BuildFlags

Description	This section shows the build tool flags that were used during the creation of the module. Of specific interest are flags that define values on the compiler command line, as well as possible optimizations. Any Attributes to this element are also permitted.
Required	NO
Data Type	Element – Complex, mixed="true"
Data Constraints	None – any attributes and/or content
Examples	`<BuildFlags MSVS=”8.0”> /EXPORT:InitializeDriver=$(IMAGE_ENTRY_POINT) / ALIGN:4096 /SUBSYSTEM:CONSOLE </BuildFlags>`

6.8 ModuleSurfaceArea.PackageDependencies

The following is the description of the ModuleSurfaceArea.PackageDependencies instance:

```
<PackageDependencies>
  <Package
    SupArchList=" ArchListType " {0,1}
    FeatureFlag=" xs:normalizedString " {0,1} >
      <Description> ... </Description> {0,1}
      <GUID
        Version=" xs:decimal " {0,1} >
          RegistryFormatGuid
      </GUID> {1}
  </Package> {1,}
</PackageDependencies>
```

Table 224. ModuleSurfaceArea.PackageDependencies

Description	This element contains a list of all packages a module depends on.
Required	NO
Data Type	Element – Complex
Data Constraints	N/A
Examples	N/A

6.8.1 PackageDependencies.Package

Table 225. PackageDependencies.Package

Description	This element specifies a package required by this module to properly build or function.
Required	YES
Data Type	Element – Complex
Data Constraints	N/A
Examples	N/A

6.8.2 PackageDependencies.Package:SupArchList

Table 226. PackageDependencies.Package:SupArchList

Description	Used to restrict the set of CPU Architectures that are allowed to use this package. If this attribute is not specified, then this package may be used with any CPU Architecture. If this attribute is specified, then only those modules that support a subset of the CPU architectures specified by this element may use this package.
Required	NO
Data Type	Attribute – ArchListType
Data Constraints	If specified, must contain one or more of the supported CPU architectures separated by spaces. The supported CPU architectures include **IA32**, **X64**, **IPF** and **EBC**

Examples	**SupArchList=”IA32”** **SupArchList=”IA32 X64”** **SupArchList=”EBC IPF X64”**

6.8.3 PackageDependencies.Package:FeatureFlag

Table 227. PackageDependencies.Package:FeatureFlag

Description	Used to restrict the use of this package. This attribute is a Boolean expression containing PCD Feature Flag names and operators. This is required to be an in-fix logical expression, evaluated left to right, for a range of values, using Relational, Equality and Logical Operators (NOT, OR, AND, LT, LE, EQ, GT, GE and XOR) that must evaluate to either True or False. The forms PcdCName and/or PcdTokenSpaceGuidCName.PcdCName are permitted; if only the PcdCName is specified, then the PcdTokenSpaceGuidCName is the same as this PCD's TokenSpaceGuidCName. Parentheses are recommended for clarity. If this attribute is specified, then this package may only be used if the Boolean expression evaluates to **true.**
Required	NO
Data Type	Attribute – xs:normalizedString
Data Constraints	PCD C names, true, false and/or Operators : NOT, AND, OR, XOR, EQ, LT, LE, GT and GE.
Examples	**FeatureFlag="true"**

6.8.4 PackageDependencies.Package.Description

Table 228. PackageDependencies.Package.Description

Description	This element describes the reason for the dependency.
Required	NO
Data Type	Element – xs:string
Data Constraints	A string.
Examples	**<Description> Requires Status Code Library Functions </Description>**

6.8.5 PackageDependencies.Package.GUID

Table 229. PackageDependencies.Package.GUID

Description	This element specifies the GUID of a package required by this module to properly build or function.
Required	YES
Data Type	Element – RegistryFormatGuid
Data Constraints	A string that represents a GUID in registry format.
Examples	**<GUID>5e0e9358-46b6-4ae2-8218-4ab8b9bbdcec</GUID>** **<GUID>b6ec423c-21d2-490d-85c6-dd5864eaa674</GUID>**

6.8.6 PackageDependencies.Package.GUID:Version

Table 230. PackageDependencies.Package.GUID:Version

Description	This attribute specifies the version of a package required by this module to properly build or function. If this attribute is listed, ONLY the specific GUID/Version package can be used. If this attribute is not specified, the highest version of a package can be used.
Required	NO
Data Type	Attribute – xs:decimal
Data Constraints	[0-9]+((\.)?[0-9]+)*
Examples	`Version="1.0"` `Version="1.10"`

6.9 ModuleSurfaceArea.Guids

It is permissible for a module to list a GUID twice, once for CONSUMES and another for PRODUCES, if and only if the GUID type is transformed by the module.

The following is the description of the **`ModuleSurfaceArea.Guids`** instance:

```
<Guids>
  <GuidCName> ... </GuidCName> {1,}
</Guids>
```

Table 231. ModuleSurfaceArea.Guids

Description	This is a list of Guids produced or consumed by this module.
Required	NO
Data Type	Element – Complex
Data Constraints	N/A
Examples	N/A

6.9.1 Guids.GuidCName

```
<GuidCName
  Usage=" UsageType " {1}
  GuidType=" GuidListType " {1}
  SupArchList=" ArchListType " {0,1}
  FeatureFlag=" xs:normalizedString " {0,1} >
    <CName> xs:NCName </CName> {1}
    <VariableName> xs:normalizedString </VariableName> {1}
    <HelpText
      Lang=" xs:language " {0,1} >
        xs:string
    </HelpText> {0,}
</GuidCName>
```

Table 232. Guids.GuidCName

Description	This element specifies information about Guids used by, or produced by this module. The GUID CName specified in this section must be defined in a package declaration. Protocols and PPIs do not need to be listed in this section.
Required	YES
Data Type	Element – Complex
Data Constraints	N/A
Examples	N/A

6.9.2 Guids.GuidCName

Table 233. Guids.GuidCName

Description	This element specifies information about Guids used by, or produced by this module. The GUID specified in this section must be defined in a package declaration file.
Required	YES
Data Type	Element – Complex
Data Constraints	N/A

Examples	N/A

6.9.3 Guids.GuidCName:Usage

Table 234. Guids.GuidCName:Usage

Description	This element specifies the GUID Usage.
Required	YES
Data Type	Attribute – UsageType
Data Constraints	One of the enumerated values: **CONSUMES**, **PRODUCES**, **SOMETIMES_CONSUMES** or **SOMETIMES_PRODUCES** **PRODUCES** means that a module will produce a GUID that does not fit into the defined PROTOCOL or PPI types. **CONSUMES** means that a module may use this GUID that does not fit into the defined PROTOCOL or PPI types.
Examples	**Usage="PRODUCES"** **Usage="CONSUMES"**

Parameters

CONSUMES

The module will use the named GUID. For GUID type (specified in the Type attribute) of:

- Event - CONSUMES means that the module has an event waiting to be signaled (i.e., the module registers a notification function and calls the function when it is signaled.

  System Table - this means that the module will use a GUIDed entry in the system table.
- EFI Variable - this means that the module may use the variable entry.
- Formset - the formset may be registered into HII by this module.
- TokenSpaceGuid - this means that an Token Space GUID will be required for PCD entries used in this module.
- Hob - this means that a HOB may need to be present in the system.
- File/FV - this means that a file must be present in an FV, such as a module that loads a processor microcode patch file.
- GUID - this means that a module may use this GUID that does not fit into the defined GUID types.

PRODUCES

This module always produces a named GUID. For GUID type (specified in the Type attribute) of:

- Boot Modes - this means that the module will change the boot mode.
- Event - this means that module will signal all events in an event group.
- System Table - this means that the module will produce a GUIDed entry in the system table.

- EFI Variable - this means that the module will write the variable.
- Formset - PRODUCES is not valid for this GUID type.
- TokenSpaceGuid - PRODUCES is not valid for this GUID type.
- Hob - this means that the HOB will be produced by the module.
- File/FV - this means that a module creates a file that is present in an FV, such as a file that contains a microcode patch.
- GUID - this means that a module will produce a GUID that does not fit into the defined PROTOCOL, PPI or GUID types.

6.9.4 Guids.GuidCName:GuidType

Table 235. Guids.GuidCName:GuidType

Description	This element specifies the type of GUID.
Required	YES
Data Type	Attribute – GuidListType
Data Constraints	One or more of the enumerated values: **`Event`**, **`File`**, **`FV`**, **`GUID`**, **`HII`**, **`HOB`**, **`SystemTable`**, **`TokenSpaceGuid`** or **`Variable`**.
Examples	**`GuidType="Event"`** **`GuidType="TokenSpaceGuid"`**

Parameters

`Event`

The functions that make up the Event used during preboot to create, close, signal, and wait for events as defined in the UEFI specification.

`SystemTable`

As defined in the PI Specification, the UEFI System Table is passed to every executable image in the DXE phase. The UEFI System Table contains a pointer to the following:

"UEFI Boot Services Table"

"UEFI Runtime Services Table"

It also contains pointers to the console devices and their associated I/O protocols. In addition, the UEFI System Table contains a pointer to the UEFI Configuration Table, which contains a list of GUID/pointer pairs. The UEFI Configuration Table may include tables such as the "DXE Services Dependencies", HOB list, ACPI table, SMBIOS table, and SAL System table.

The UEFI Boot Services Table contains services to access the contents of the handle database. The handle database is where protocol interfaces produced by drivers are registered. Other drivers can use the UEFI Boot Services to look up these services produced by other drivers.

Variable

The UEFI and PI specifications define UEFI Variables as key/value pairs that consist of identifying information plus attributes (the key) and arbitrary data (the value). Variables are intended for use as a means to store data that is passed between the UEFI environment implemented in the platform and UEFI OS loaders and other applications that run in the UEFI environment.

HII

These GUIDs refer to the core code and services that are required for an implementation of the Human Interface Infrastructure (HII).

TokenSpaceGuid

These GUIDs are used as a definition of a group of Platform Configuration Database (PCD) entries. Different packages of PCDs are identified by a Token Space GUID. Within each of these packages, the individual PCD's Token number is unique. PCDs are then identified by the Token Space GUID and the PCD's Token number.

HOB

These GUIDs are for manipulation of the PEI-DXE Hand Off Blocks. The HOB types are defined in the PI Specification.

GUID

These are GUIDs that do not fall into PPI, PROTOCOL or any of the above definitions.

6.9.5 Guids.GuidCName:SupArchList

Table 236. Guids.GuidCName:SupArchList

Description	Used to restrict the set of CPU Architectures that are allowed to use this GUID. If this attribute is not specified, then this GUID may be used with any CPU Architecture. If this attribute is specified, then only those modules that support a subset of the CPU architectures specified by this element may use this GUID.
Required	NO
Data Type	Attribute – ArchListType
Data Constraints	If specified, must contain one or more of the supported CPU architectures separated by spaces. The supported CPU architectures include **IA32**, **X64**, **IPF** and **EBC**
Examples	**SupArchList="IA32"** **SupArchList="IA32 X64"** **SupArchList="EBC IPF X64"**

6.9.6 Guids.GuidCName:FeatureFlag

Table 237. Guids.GuidCName:FeatureFlag

Description	Used to restrict the use of this GUID. This attribute is a Boolean expression containing PCD Feature Flag names and operators. This is required to be an in-fix logical expression, evaluated left to right, for a range of values, using Relational, Equality and Logical Operators (NOT, OR, AND, LT, LE, EQ, GT, GE and XOR) that must evaluate to either True or False. The forms PcdCName and/or PcdTokenSpaceGuidCName.PcdCName are permitted; if only the PcdCName is specified, then the PcdTokenSpaceGuidCName is the same as this PCD's TokenSpaceGuidCName. Parentheses are recommended for clarity. If this attribute is specified, then this GUID may only be used if the Boolean expression evaluates to **true**.
Required	NO
Data Type	Attribute – xs:normalizedString
Data Constraints	PCD C names, true, false and/or Operators : NOT, AND, OR, XOR, EQ, LT, LE, GT and GE.
Examples	**FeatureFlag="true"**

6.9.7 Guids.GuidCName.CName

Table 238. Guids.GuidCName.CName

Description	This element specifies the C name of the GUID.
Required	YES
Data Type	Element – xs:NCName
Data Constraints	Must be a valid GUID C name

Examples	`<CName> gEfiEventReadyToBootGuid </CName>` `<CName>gEfiConsoleInDeviceGuid</CName>`

6.9.8 Guids.GuidCName.VariableName

Table 239. Guids.GuidCName.VariableName

Description	This element specifies the Variable name for a Variable GUID.
Required	NO
Data Type	Element – xs:normalizedString
Data Constraints	Must be a valid Variable Name in either Unicode, Hex Array or L"string" (C style.) This element is only valid for an EFI Variable GUID type.
Examples	`<VariableName> L"Reset" </VariableName>` `<VariableName> 0x00 0x52 0x00 0x65 0x00 0x73 0x00 0x65 0x00 0x74 </VariableName>`

6.9.9 Guids.GuidCName.HelpText

Table 240. Guids.GuidCName.HelpText

Description	This element specifies the HelpText describing correct usage of the GUID
Required	NO
Data Type	Element – xs:string
Data Constraints	A string that contains one or more lines of text.
Examples	`<HelpText> Check the device handle, if it is a hot plug device, do not put the device path into ConInDev, and install gEfiConsoleInDeviceGuid to the device handle directly.` `The policy is, make hot plug device plug in and play immediately. </HelpText>`

6.9.10 Guids.GuidCName.HelpText:Lang

Table 241. Guids.GuidCName.HelpText:Lang

Description	The language code uses an RFC 1766 language code identifier
Required	NO
Data Type	Attribute – xs:language
Data Constraints	If present, the language code must use an RFC 1766 language code identifier. If not specified, the default is "en-us.'
Examples	`Lang="en-us"`

6.10 ModuleSurfaceArea.Protocols

The following is the description of the `ModuleSurfaceArea.Protocols` instance:

```
<Protocols>
  <Protocol
    Usage=" ProtocolUsageType " {1}
    Notify=" xs:boolean " {0,1}
    SupArchList=" ArchListType " {0,1}
    FeatureFlag=" xs:normalizedString " {0,1} >
      <CName> xs:NCName </CName> {1}
      <HelpText
         Lang=" xs:language " {0,1} >
           xs:string
      </HelpText> {0,}
  </Protocol> {1,}
</Protocols>
```

Table 242. ModuleSurfaceArea.Protocols

Description	This element contains a list of all Protocols a module depends on or produces.
Required	NO
Data Type	Element – Complex
Data Constraints	N/A
Examples	N/A

6.10.1 Protocols.Protocol

Table 243. Protocols.Protocol

Description	This element specifies the C name of a Protocol required by this module to properly build or function, or produced by this module.
Required	YES
Data Type	Element – ComplexType
Data Constraints	Only child elements are allowed within the element
Examples	N/A

6.10.2 Protocols.Protocol:Usage

Table 244. Protocols.Protocol:Usage

Description	This attribute specifies whether the Protocol is needed or produced by the module.
Required	YES
Data Type	Attribute – ProtocolUsageType
Data Constraints	One of the enumerated values: **CONSUMES**, **PRODUCES**, **SOMETIMES_CONSUMES**, **SOMETIMES_PRODUCES**, **TO_START** or **BY_START**.

Examples	**Usage="PRODUCES"** **Usage="CONSUMES"**

Parameters

CONSUMES

This module does not install the protocol, but needs to locate a protocol. Not valid if the Notify attribute is true.

PRODUCES

This module will install this protocol. Not valid if the Notify attribute is true.

SOMETIMES_CONSUMES

This module does not install the protocol, but may need to locate a protocol under certain conditions, (such as if it is present.) If the Notify attribute is set, then the module will use the protocol, named by GUID, via a registry protocol notify mechanism.

SOMETIMES_PRODUCES

This module will install this protocol under certain conditions. Not valid if the Notify attribute is true.

TO_START

The protocol is consumed by a Driver Binding protocol Start function. Thus the protocol is used as part of the UEFI driver model. Not valid if the Notify attribute is true.

BY_START

The protocol is produced by a Driver Binding protocol Start function. Thus the protocol is used as part of the UEFI driver model. Not valid if the Notify attribute is true.

6.10.3 Protocols.Protocol:Notify

Table 245. Protocols.Protocol:Notify

Description	This attribute specifies whether this is a Protocol or ProtocolNotify. If set, then the module will use this protocol, named by GUID, via a registry protocol notify mechanism.
Required	NO
Data Type	Attribute – xs:boolean
Data Constraints	Only required for ProtocolNotify, in which the value must be set to true. The default (non-specified) value is false.
Examples	**Notify="true"**

6.10.4 Protocols.Protocol:SupArchList

Table 246. Protocols.Protocol:SupArchList

Description	Used to restrict the set of CPU Architectures that are allowed to use this protocol. If this attribute is not specified, then this protocol may be used with any CPU Architecture. If this attribute is specified, then only those modules that support a subset of the CPU architectures specified by this element may use this protocol.
Required	NO
Data Type	Attribute – ArchListType
Data Constraints	If specified, must contain one or more of the supported CPU architectures separated by spaces. The supported CPU architectures include **IA32**, **X64**, **IPF** and **EBC**
Examples	**SupArchList="IA32"** **SupArchList="IA32 X64"** **SupArchList="EBC IPF X64"**

6.10.5 Protocols.Protocol:FeatureFlag

Table 247. Protocols.Protocol:FeatureFlag

Description	Used to restrict the use of this protocol. This attribute is a Boolean expression containing PCD Feature Flag names and operators. This is required to be an in-fix logical expression, evaluated left to right, for a range of values, using Relational, Equality and Logical Operators (NOT, OR, AND, LT, LE, EQ, GT, GE and XOR) that must evaluate to either True or False. The forms PcdCName and/or PcdTokenSpaceGuidCName.PcdCName are permitted; if only the PcdCName is specified, then the PcdTokenSpaceGuidCName is the same as this PCD's TokenSpaceGuidCName. Parentheses are recommended for clarity. If this attribute is specified, then this protocol may only be used if the Boolean expression evaluates to **true**.
Required	NO
Data Type	Attribute – xs:normalizedString

Data Constraints	PCD C names, true, false and/or Operators : NOT, AND, OR, XOR, EQ, LT, LE, GT and GE.
Examples	`FeatureFlag="false"`

6.10.6 Protocols.Protocol.CName

Table 248. Protocols.Protocols.CName

Description	This element specifies the C name of the Protocol.
Required	YES
Data Type	Element – xs:NCName
Data Constraints	Must be a valid GUID C name
Examples	`<CName>gEfiBlockIoProtocolGuid</CName>` `<CName> gEfiBlockIoProtocolGuid </CName>`

6.10.7 Protocols.Protocol.HelpText

Table 249. Protocols.Protocol.HelpTex

Description	This element specifies the HelpText describing correct usage of the Protocol
Required	NO
Data Type	Element – xs:string
Data Constraints	A string that contains one or more lines of text.
Examples	`<HelpText> The Block I/O Protocol provides services to read, write, and flush block to a block oriented storage device such as a hard disk, CD-ROM, DVD, and floppy. This Protocol is available to all CPU types, but is restricted for use by DXE_DRIVER, UEFI_DRIVER, and UEFI_APPLICATION module types. </HelpText>`

6.10.8 Protocols.Protocol.HelpText:Lang

Table 250. Protocols.Protocol.HelpText:Lang

Description	The language code uses an RFC 1766 language code identifier
Required	NO
Data Type	Attribute – xs:language
Data Constraints	If present, the language code must use an RFC 1766 language code identifier. If not specified, the default is “en-us.’
Examples	`Lang=”en-us”`

6.11 ModuleSurfaceArea.PPIs

The following is the description of the `ModuleSurfaceArea.PPIs` instance:

```
<PPIs>
  <Ppi
    Usage=" UsageType " {1}
    Notify=" xs:boolean " {0,1}
    SupArchList=" ArchListType " {0,1}
    FeatureFlag=" xs:normalizedString " {0,1} >
      <CName> xs:NCName </CName> {1}
      <HelpText
         Lang=" xs:language " {0,1} >
           xs:string
      </HelpText> {0,}
  </Ppi> {1,}
</PPIs>
```

Table 251. ModuleSurfaceArea.PPIs

Description	This element contains a list of all PPIs a module depends on or produces.
Required	NO
Data Type	Element – Complex
Data Constraints	N/A
Examples	N/A

6.11.1 PPIs.Ppi

Table 252. PPIs.Ppi

Description	This element specifies the C name of a Ppi required by this module to properly build or function, or produced by this module.
Required	YES
Data Type	Element – ComplexType
Data Constraints	N/A
Examples	N/A

6.11.2 PPIs.Ppi:Usage

Table 253. PPIs.Ppi:Usage

Description	This attribute specifies whether the Ppi is required or produced by the module.
Required	YES
Data Type	Attribute – UsageType
Data Constraints	One of the enumerated values: **CONSUMES**, **PRODUCES**, **SOMETIMES_CONSUMES** or **SOMETIMES_PRODUCES**.
Examples	**Usage="PRODUCES"** **Usage="CONSUMES"**

Parameters

CONSUMES

This module does not install the PPI, but needs to locate a PPI. Not valid if the Notify attribute is true.

PRODUCES

This module will load this PPI. Not valid if the Notify attribute is true.

SOMETIMES_CONSUMES

This module does not install the PPI, but may need to locate a PPI under certain conditions or execution paths. If the Notify attribute is set, then the module will use the PPI, named by GUID, via a registry PPI notify mechanism.

SOMETIME_PRODUCES

This module will load this PPI under certain conditions or execution paths. Not valid if the Notify attribute is true.

6.11.3 PPIs.Ppi:Notify

Table 254. Ppis.Ppi:Notify

Description	This attribute specifies whether this is a Ppi or PpiNotify. If set to, the module requires or consumes a PPI, named by GUID, via a register PPI notify mechanism.
Required	NO
Data Type	Attribute – xs:boolean
Data Constraints	Only required for PpiNotify, in which the value must be set to true. The default (non-specified) value is false.
Examples	**`Notify="true"`**

6.11.4 PPIs.Ppi:SupArchList

Table 255. PPIs.Ppi:SupArchList

Description	Used to restrict the set of CPU Architectures that are allowed to use this Ppi. If this attribute is not specified, then this Ppi may be used with any CPU Architecture. If this attribute is specified, then only those modules that support a subset of the CPU architectures specified by this element may use this Ppi.
Required	NO
Data Type	Attribute – ArchListType
Data Constraints	If specified, must contain one or more of the supported CPU architectures separated by spaces. The supported CPU architectures include **`IA32`**, **`X64`**, **`IPF`** and **`EBC`**
Examples	**`SupArchList="IA32"`** **`SupArchList="IA32 X64"`** **`SupArchList="EBC IPF X64"`**

6.11.5 PPIs.Ppi:FeatureFlag

Table 256. PPIs.Ppi:FeatureFlag

Description	Used to restrict the use of this Ppi. This attribute is a Boolean expression containing PCD Feature Flag names and operators. This is required to be an in-fix logical expression, evaluated left to right, for a range of values, using Relational, Equality and Logical Operators (NOT, OR, AND, LT, LE, EQ, GT, GE and XOR) that must evaluate to either True or False. The forms PcdCName and/or PcdTokenSpaceGuidCName.PcdCName are permitted; if only the PcdCName is specified, then the PcdTokenSpaceGuidCName is the same as this PCD's TokenSpaceGuidCName. Parentheses are recommended for clarity. If this attribute is specified, then this Ppi may only be used if the Boolean expression evaluates to **`true`**.
Required	NO
Data Type	Attribute – xs:normalizedString

Data Constraints	PCD C names, true, false and/or Operators : NOT, AND, OR, XOR, EQ, LT, LE, GT and GE.
Examples	`FeatureFlag="false"`

6.11.6 PPIs.Ppi.CName

Table 257. PPIs.Ppi.CName

Description	This element specifies the C name of the Ppi.
Required	YES
Data Type	Element – xs:NCName
Data Constraints	Must be a valid C name.
Examples	`<CName>gEfiDxeIplPpiGuid</CName>` `<CName>gEfiPeiStallPpiGuid</CName>`

6.11.7 PPIs.Ppi.HelpText

Table 258. PPIs.Ppi.HelpTex

Description	This element specifies the HelpText describing correct usage of the Ppi
Required	NO
Data Type	Element – xs:string
Data Constraints	A string that contains one or more lines of text.
Examples	`<HelpText> The Block I/O Ppi provides services to read, write, and flush block to a block oriented storage device such as a hard disk, CD-ROM, DVD, and floppy. This Ppi is available to all CPU types, but is restricted for use by DXE_DRIVER, UEFI_DRIVER, and UEFI_APPLICATION module types.</HelpText>`

6.11.8 PPIs.Ppi.HelpText:Lang

Table 259. PPIs.Ppi.HelpText:Lang

Description	The language code uses an RFC 1766 language code identifier
Required	NO
Data Type	Attribute – xs:language
Data Constraints	If present, the language code must use an RFC 1766 language code identifier. If not specified, the default is “en-us.’
Examples	`Lang=”en-us”`

6.12 ModuleSurfaceArea.Externs

The following is the description of the `ModuleSurfaceArea.Externs` instance:

```
<Externs>
  <Extern
    SupArchList=" ArchListType " {0,1}
    FeatureFlag=" xs:normalizedString " {0,1} >
      Start Choice
        Start Sequence
          <EntryPoint> xs:NCName </EntryPoint> {0,1}
          <UnloadImage> xs:NCName </UnloadImage> {0,1}
        End Sequence
        Start Sequence
          <Constructor> xs:NCName </Constructor> {0,1}
          <Destructor> xs:NCName </Destructor> {0,1}
        End Sequence
      End Choice
      <HelpText
         Lang=" xs:language " {0,1} >
           xs:string
      </HelpText> {0,}
  </Extern> {1,}
</Externs>
```

Table 260. ModuleSurfaceArea.Externs

Description	This section defines additional information about dirvers and libraries.
Required	NO
Data Type	Element – Complex
Data Constraints	N/A
Examples	N/A

6.12.1 Externs.Extern

Table 261. Externs.Extern

Description	These elements specify multiple setions of additional information.
Required	NO
Data Type	Element – Complex
Data Constraints	Multiple Instances of Extern Statements are permitted, and at least one of the following sub-elements must be specified.
Examples	N/A

6.12.2 Externs.Extern:SupArchList

Table 262. Externs.Extern:SupArchList

Description	Used to restrict the set of CPU Architectures that are allowed to use this Extern. If this attribute is not specified, then this Extern may be used with any CPU Architecture. If this attribute is specified, then only those modules that support a subset of the CPU architectures specified by this element may use this Extern.
Required	NO
Data Type	Attribute – ArchListType
Data Constraints	If specified, must contain one or more of the supported CPU architectures separated by spaces. The supported CPU architectures include **IA32**, **X64**, **IPF** and **EBC**
Examples	**SupArchList="IA32"** **SupArchList="IA32 X64"** **SupArchList="EBC IPF X64"**

6.12.3 Externs.Extern:FeatureFlag

Table 263. Externs.Extern:FeatureFlag

Description	Used to restrict the use of this Extern. This attribute is a Boolean expression containing PCD Feature Flag names and operators. This is required to be an in-fix logical expression, evaluated left to right, for a range of values, using Relational, Equality and Logical Operators (NOT, OR, AND, LT, LE, EQ, GT, GE and XOR) that must evaluate to either True or False. The forms PcdCName and/or PcdTokenSpaceGuidCName.PcdCName are permitted; if only the PcdCName is specified, then the PcdTokenSpaceGuidCName is the same as this PCD's TokenSpaceGuidCName. Parentheses are recommended for clarity. If this attribute is specified, then this Extern may only be used if the Boolean expression evaluates to **true**.
Required	NO
Data Type	Attribute – xs:normalizedString
Data Constraints	PCD C names, true, false and/or Operators : NOT, AND, OR, XOR, EQ, LT, LE, GT and GE.
Examples	**FeatureFlag="true"**

6.12.4 Externs.Extern.EntryPoint

Table 264. Externs.Extern.EntryPoint

Description	This element specifies the Module Entry Point of the Module. C name of the module entry point function. The PE32 image entry point goes into a library and the library calls this entry point in the module.
Required	NO
Data Type	Element – xs:NCName
Data Constraints	Must be a valid C name. Valid for drivers only.

Examples	**<EntryPoint> GenericMemoryTestEntryPoint </EntryPoint>** **<EntryPoint>BarEntry</EntryP**oint>

6.12.5 Externs.Extern.UnloadImage

Table 265. Externs.Extern.UnloadImage

Description	This element specifies the Module Unload Image of the Module. C name of the module unload function. This function is optional as a module is not required to support unload.
Required	NO
Data Type	Element – xs:NCName
Data Constraints	Must be a valid C name. Valid for drivers only.
Examples	`<UnloadImage>FooUnload</UnloadImage>` `<UnloadImage>BarUnload</UnloadImage>`

6.12.6 Externs.Extern.Constructor

Table 266. Externs.Extern.Constructor

Description	This element specifies the Constructor of a Library. While most library instances do not produce Constructors, those that do, should include the C name here. Library constructors are called before the module entry point function is executed. Any library that performs initialization (such as the Dxe SMM report status code library) should publish a constructor.
Required	NO
Data Type	Element – xs:NCName
Data Constraints	Must be a valid C name. Valid for a Library Module
Examples	`<Constructor>BarInit</Constructor>` `<Constructor>FooInit</Constructor>`

6.12.7 Externs.Extern.Destructor

Table 267. Externs.Extern.Destructor

Description	This element specifies the Destructor of a Library. While most library instances do not produce Destructors, those that do, should include the C name here. Library destructors are called after the module entry point function is executed. Any library that performs a cleanup (such as the UEFI Runtime library) should publish a destructor.
Required	NO
Data Type	Element – xs:NCName
Data Constraints	Must be a valid C name. Valid for a Library Module
Examples	`<Destructor>BarKill</Destructor>` `<Destructor>FooKill</Destructor>`

6.12.8 Externs.Extern.HelpText

Table 268. Externs.Extern.HelpText

Description	This element specifies the HelpText describing the module EntryPoint, UnloadImage, Constructor or Destructor.
Required	NO
Data Type	Element – xs:string
Data Constraints	A string that contains one or more lines of text.
Examples	`<HelpText> GenericMemoryTestEntryPoint is the generic memory test driver's entry point, it can initialize private data to default value. </HelpText>`

6.12.9 Externs.Extern.HelpText:Lang

Table 269. Externs.Extern.HelpText:Lang

Description	The language code uses an RFC 1766 language code identifier
Required	NO
Data Type	Attribute – xs:language
Data Constraints	If present, the language code must use an RFC 1766 language code identifier. If not specified, the default is “en-us.’
Examples	`Lang=”en-us”`

6.13 ModuleSurfaceArea.PcdCoded

Note that if the Binary attribute was set for the ModuleSurfaceArea, then only include the PCDs that are of type: PatchPcd, Pcd and/or PcdEx. The "DefaultValue" becomes the real value used during the build of the module in the context of a platform and must be set. The PcdItemType must be set to the way the platform used the PCD. If a binary is present, along with the sources, and the Binary attribute is NOT set, then this is a standard section, and the Pcd sections of the AsBuilt must be used to show how the binary was created.

The following is the description of the **`ModuleSurfaceArea.PcdCoded`** instance:

```
<PcdCoded>
  <PcdEntry
    PcdItemType=" PcdItemTypes " {1}
    PcdUsage=" xs:NCName " {1}
    SupArchList=" ArchListType " {0,1}
    FeatureFlag=" xs:normalizedString " {0,1} >
      <CName> xs:NCName </CName> {1}
     <TokenSpaceGuidCName> xs:NCName </TokenSpaceGuidCName> {1}
      <DefaultValue> xs:normalizedString </DefaultValue> {0,1}
      <HelpText
         Lang=" xs:language " {0,1} >
           xs:string
      </HelpText> {0,}
  </PcdEntry> {1,}
</PcdCoded>
```

Table 270. ModuleSurfaceArea.PcdCoded

Description	The list of PCDs the module was coded to use.
Required	NO
Data Type	Element - Complex
Data Constraints	N/A
Examples	N/A

6.13.1 PcdCoded.PcdEntry

Table 271. ModuleSurfaceArea.PcdCoded.PcdEntry

Description	Defines how the module was coded for this PCD Entry.
Required	YES
Data Type	Element - Complex
Data Constraints	N/A
Examples	N/A

6.13.2 PcdCoded.PcdEntry:PcdItemType

Table 272. PcdCoded.PcdEntry:PcdItemType

Description	Define the PCD type for this entry.
Required	YES
Data Type	Attribute – PcdItemTypes
Data Constraints	Must contain one of the following Enumerated Data Types: **FeaturePcd**, **FixedPcd**, **PatchPcd**, **Pcd** or **PcdEx**.

Examples	**PcdItemType="FixedPcd"** **PcdItemType="FixedPcd PatchPcd"** **PcdItemType="Pcd"**

Parameters

FeaturePcd

The FeaturePcd is used as a boolean flag to enable or disable specific features.

FixedPcd

The FixedPcd is used to flag the platform integrator that the module requires a static mapping of the PCD.

PatchPcd

The PatchPcd is used to flag the platform integrator that the module requires a mapping of the PCD where the value can be altered in the binary (the value needs to be a known offset,, with information about data size and type available.

Pcd

The Pcd is used to flag the platform integrator that the module may be used as a Fixed Pcd, Patch Pcd, or a Dynamic PCD - where the value is registered in a PCD database within the system and the value can be found/set using either a PCD PPI or PCD Protocol driver. The platform integrator should select how he wants to use the PCD.

PcdEx

The PcdEx is used to flag the platform integrator that the module is coded for a Dynamic PCD - where the value is registered in a PCD database within the system and the value can be found/set using either a PCD PPI or PCD Protocol driver.

6.13.3 PcdCoded.PcdEntry:PcdUsage

Table 273. PcdCoded.PcdEntry:PcdUsage

Description	This attribute specifies whether the Pcd is required or produced by the module.
Required	YES
Data Type	Attribute – xs:NCName
Data Constraints	One of the enumerated values: **CONSUMES**, **SOMETIMES_CONSUMES**, **PRODUCES** or **SOMETIMES_PRODUCES**. For PcdItemTypes of FeatureFlag, this entry must be **PRODUCES**.
Examples	**PcdUsage="PRODUCES"** **PcdUsage="CONSUMES"**

Parameters

CONSUMES

This module always gets the PCD entry.

PRODUCES

The module always sets the PCD entry.

SOMETIMES_CONSUMES

The module gets the PCD entry under certain conditions or execution paths.

SOMETIMES_PRODUCES

The module sets the PCD entry under certain conditions or execution paths.

6.13.4 PcdCoded.PcdEntry:SupArchList

Table 274. PcdCoded.PcdEntry:SupArchList

Description	Used to restrict the set of CPU Architectures that are allowed to use this PCD. If this attribute is not specified, then this PCD may be used with any CPU Architecture. If this attribute is specified, then only those modules that support a subset of the CPU architectures specified by this element may use this PCD.
Required	NO
Data Type	Attribute – ArchListType
Data Constraints	If specified, must contain one or more of the supported CPU architectures separated by spaces. The supported CPU architectures include **IA32**, **X64**, **IPF** and **EBC**
Examples	**SupArchList="IA32"** **SupArchList="IA32 X64"** **SupArchList="EBC IPF X64"**

6.13.5 PcdCoded.PcdEntry:FeatureFlag

Table 275. PcdCoded.PcdEntry:FeatureFlag

Description	Used to restrict the use of this PCD. This attribute is a Boolean expression containing PCD Feature Flag names and operators. This is required to be an in-fix logical expression, evaluated left to right, for a range of values, using Relational, Equality and Logical Operators (NOT, OR, AND, LT, LE, EQ, GT, GE and XOR) that must evaluate to either True or False. The forms PcdCName and/or PcdTokenSpaceGuidCName.PcdCName are permitted; if only the PcdCName is specified, then the PcdTokenSpaceGuidCName is the same as this PCD's TokenSpaceGuidCName. Parentheses are recommended for clarity. If this attribute is specified, then this PCD may only be used if the Boolean expression evaluates to **true**.
Required	NO
Data Type	Attribute – xs:normalizedString
Data Constraints	PCD C names, true, false and/or Operators : NOT, AND, OR, XOR, EQ, LT, LE, GT and GE.
Examples	**FeatureFlag="true"**

6.13.6 PcdCoded.PcdEntry.CName

Table 276. PcdCoded.PcdEntry.CName

Description	This element specifies the C name of the PCD entry
Required	YES
Data Type	Element – xs:NCName
Data Constraints	A valid C name listed in a declaration file that declares the PCD.

Examples	`<CName>PcdMaximumUnicodeStringLength</CName>` `<CName> PcdMaximumAsciiStringLength </CName>`

6.13.7 PcdCoded.PcdEntry.TokenSpaceGuidCName

Table 277. PcdCoded.PcdEntry.TokenSpaceGuidCName

Description	This element specifies the C name of the PCD defined TokenSpaceGuidCName. The actual GUID value must be declared in a package declaration file that may not be part of this distribution.
Required	YES
Data Type	Element – xs:NCName
Data Constraints	A valid C name listed in a declaration file that declares the PCD.
Examples	`<TokenSpaceGuidCName> gEfiMdePkgTokenSpaceGuid </TokenSpaceGuidCName>`

6.13.8 PcdCoded.PcdEntry.DefaultValue

Table 278. PcdCoded.PcdEntry.DefaultValue

Description	This element specifies the C name of the PCD defined TokenSpaceGuidCName
Required	NO
Data Type	Element – xs:normalizedString
Data Constraints	Any valid string of printable characters.
Examples	`<DefaultValue>0x0048 0x0052</DefaultValue>` `<DefaultValue>LBigBear</DefaultValue>`

6.13.9 PcdCoded.PcdEntry.HelpText

Table 279. PcdCoded.PcdEntry.HelpText

Description	This element specifies the HelpText describing how the module was coded for this PCD
Required	NO
Data Type	Element – xs:string
Data Constraints	A string that contains one or more lines of text.

Examples	**<HelpText> This module uses the FixedPcd: PcdDxeIplSwitchToLongMode to compute the top of the stack we were allocated, which is used to load X64 dxe core. Pre-allocate a 32 bytes which conforms to x64 calling convention. The first four parameters to a function are passed in rcx, rdx, r8 and r9. Any further parameters are pushed on the stack. Furthermore, space (4 * 8bytes) for the register parameters is reserved on the stack, in case the called function wants to spill them; this is important if the function is variadic. </HelpText>**

6.13.10 PcdCoded.PcdEntry.HelpText:Lang

Table 280. PcdCoded.PcdEntry.HelpText:Lang

Description	The language code uses an RFC 1766 language code identifier
Required	NO
Data Type	Attribute – xs:language
Data Constraints	If present, the language code must use an RFC 1766 language code identifier. If not specified, the default is “en-us.’
Examples	**Lang=”en-us”**

6.14 ModuleSurfaceArea.PeiDepex

The following is the description of the **ModuleSurfaceArea.PeiDepex** instance:

```
<PeiDepex>
  <Expression> xs:string </Expression> {1}
  <HelpText
    Lang=" xs:language " {0,1} >
      xs:normalizedString
  </HelpText> {0,1}
</PeiDepex>
```

Table 281. ModuleSurfaceArea.PeiDepex

Description	This describes the PEIM dependency expression.
Required	NO
Data Type	Element – Complex
Data Constraints	N/A
Examples	N/A

6.14.1 PeiDepex.Expression

Table 282. PeiDepex.Expression

Description	This is a string contains valid dependency expression, processed left to right. Tools may use this data to create a PEI Depex section. The GUID value associated with the C name can be found by searching the GUID declarations in the package surface area.
Required	YES
Data Type	Element – xs:normalizedString
Data Constraints	A single line that will be used to generate a PEI dependency section.
Examples	`gEfiPeiMemoryDiscoveredPpiGuid OR gEfiTemporaryRamSupportPpiGuid`

6.14.1.1 PEIM Dependency Expression Grammar

The expression must conform to the following grammar.

Note: *Future releases of PI specification may take precedence over this grammar):*

```
<PeiDepex>   ::= | <bool>

<bool>       ::= <bool> AND <term>
                 | <bool> OR <term>
                 | <term>
<term>       ::= NOT <factor>
                 | <factor>
<factor>     ::= <bool>
                 | TRUE
                 | FALSE
                 | <GUID>
                 | END

<GUID>       ::=  {<RegistryFormatGuid>} {<GuidCName>}
```

6.14.2 PeiDepex.HelpText

Table 283. PeiDepex.HelpText

Description	This is a string contains valid dependency expression, processed left to right. Tools may use this data to create a PEI Depex section.
Required	YES
Data Type	Element – xs:normalizedString
Data Constraints	A single line that will be used to generate a PEI dependency section.
Examples	`gEfiPeiMemoryDiscoveredPpiGuid OR gEfiTemporaryRamSupportPpiGuid`

6.14.3 PeiDepex.HelpText:Lang

Table 284. PeiDepex.HelpText:Lang

Description	The language code uses an RFC 1766 language code identifier
Required	NO
Data Type	Attribute – xs:language
Data Constraints	If present, the language code must use an RFC 1766 language code identifier. If not specified, the default is “en-us.’
Examples	`Lang=”en-us”`

6.15 Module SurfaceArea.DxeDepex

The following is the description of the **`ModuleSurfaceArea.DxeDepex`** instance:

```
<DxeDepex>
  <Expression> xs:string </Expression> {1}
  <HelpText
    Lang=" xs:language " {0,1} >
      xs:normalizedString
  </HelpText> {0,1}
</DxeDepex>
```

Table 285. ModuleSurfaceArea.DxeDepex

Description	This describes the DXE driver dependency expression.
Required	NO
Data Type	Element – Complex
Data Constraints	N/A
Examples	N/A

6.15.1 DxeDepex.Expression

Table 286. DxeDepex.Expression

Description	This is a string contains valid dependency expression, processed left to right. Tools may use this data to create a Dxe Depex section.The GUID value associated with the C name can be found by searching the GUID declarations in the package surface area.
Required	YES
Data Type	Element – xs:normalizedString
Data Constraints	A single line that will be appended to a flag line for the tool code entered.
Examples	`gEfiFirmwareVolumeBlockProtocolGuid AND gEfiAlternateFvBlockGuid AND gEfiFaultTolerantWriteLiteProtocolGuid`

6.15.1.1 DXE Dependency Expression Grammar

The Expression must conform to the following grammar (note that future releases of PI specification may take precedence over this grammar):

```
<DxeDepex>   ::= BEFORE <guid>
               | AFTER <guid>
               | SOR <bool>
               | <bool>
<bool>       ::= <bool> AND <term>
               | <bool> OR <term>
               | <term>
<term>       ::= NOT <factor>
               | <factor>
<factor>     ::= <bool>
               | TRUE
               | FALSE
               | GUID
               | END
<guid>       ::= {<RegistryFormatGuid>} {<GuidCName>}
```

6.15.2 DxeDepex.HelpText

Table 287. DxeDepex.HelpText

Description	This element specifies the HelpText describing why this DXE depex is required for the module.
Required	NO
Data Type	Element – xs:string
Data Constraints	A string that contains one or more lines of text.
Examples	`<HelpText> This module required the following. </HelpText>`

6.15.3 DxeDepex.HelpText:Lang

Table 288. DxeDepex.HelpText:Lang

Description	The language code uses an RFC 1766 language code identifier
Required	NO
Data Type	Attribute – xs:language
Data Constraints	If present, the language code must use an RFC 1766 language code identifier. If not specified, the default is “en-us.’
Examples	`Lang=”en-us”`

6.16 Module SurfaceArea.SmmDepex

The following is the description of the **`ModuleSurfaceArea.SmmDepex`** instance:

```
<SmmDepex>
  <Expression> xs:string </Expression> {1}
  <HelpText
    Lang=" xs:language " {0,1} >
      xs:normalizedString
  </HelpText> {0,1}
</SmmDepex>
```

Table 289. ModuleSurfaceArea.SmmDepex

Description	This describes the SMM (DXE) driver dependency expression.
Required	NO
Data Type	Element – Complex
Data Constraints	N/A
Examples	N/A

6.16.1 SmmDepex.Expression

Table 290. SmmDepex.Expression

Description	This is a string contains valid dependency expression, processed left to right. Tools may use this data to create a SMM (Dxe) Depex section. The GUID value associated with the C name can be found by searching the GUID declarations in the package surface area.
Required	YES
Data Type	Element – xs:normalizedString
Data Constraints	A single line that will be appended to a flag line for the tool code entered.
Examples	`gEfiFirmwareVolumeBlockProtocolGuid AND gEfiAlternateFvBlockGuid AND gEfiFaultTolerantWriteLiteProtocolGuid`

6.16.1.1 SMM Dependency Expression Grammar

The Expression must conform to the following grammar.

Note: *Future releases of PI specification may take precedence over this grammar):*

```
<SmmDepex>   ::= BEFORE <guid>
               | AFTER <guid>
               | SOR <bool>
               | <bool>
<bool>       ::= <bool> AND <term>
               | <bool> OR <term>
               | <term>
<term>       ::= NOT <factor>
               | <factor>
<factor>     ::= <bool>
               | TRUE
               | FALSE
               | GUID
               | END
<guid>       ::= {<RegistryFormatGuid>} {<GuidCName>}
```

6.16.2 SmmDepex.HelpText

Table 291. SmmDepex.HelpText

Description	This element specifies the HelpText describing why this SMM depex is required for the module.
Required	NO
Data Type	Element – xs:string
Data Constraints	A string that contains one or more lines of text.
Examples	`<HelpText> This module required the following. </HelpText>`

6.16.3 SmmDepex.HelpText:Lang

Table 292. SmmDepex.HelpText:Lang

Description	The language code uses an RFC 1766 language code identifier
Required	NO
Data Type	Attribute – xs:language
Data Constraints	If present, the language code must use an RFC 1766 language code identifier. If not specified, the default is “en-us.’
Examples	`Lang=”en-us”`

6.17 ModuleSurfaceArea.MiscellaneousFiles

The following is the description of the **`ModuleSurfaceArea.MiscellaneousFiles`**

instance:

```
<MiscellaneousFiles>
  <Description> xs:string </Description> {0,1}
  <Filename
    Executable=" xs:boolean " {0,1}
      xs:anyURI
  </Filename> {1,}
</MiscellaneousFiles>
```

Table 293. ModuleSurfaceArea.MiscellaneousFiles

Description	This element contains any module specific miscellaneous files not defined previously.
Required	NO
Data Type	Element – Complex
Data Constraints	Free-form text or any well formed XML element(s.)
Examples	N/A

6.17.1 MiscellaneousFiles.Description

Table 294. MiscellaneousFiles.Description

Description	Any information that might be appropriate for files listed in this section.
Required	NO
Data Type	Element – xs:string
Data Constraints	A string that starts with a letter followed by any combination of letters, digits, underscores, and periods. Whitespace characters are allowed.
Examples	`<Description> Readme.txt is a quick start guide for using this module. </Description>`

6.17.2 MiscellaneousFiles.Filename

Table 295. MiscellaneousFiles.Filename

Description	A file that is included in the distribution package that does not pertain to any of the previously defined sections.
Required	NO
Data Type	Element – xs:anyURI
Data Constraints	The path and filename of the file, relative to the Module's "root" directory (as specified in theDistribution Content File.)
Examples	`<File>FooBarMfgBuildReadMe.txt</File>` `<File> Specs/FooBar_Rules.pdf </File>`

6.17.3 Misc.Filename:Executable

Table 296. Misc.Filename:Executable

Description	This flag is used during installation to ensure that the file is executable.
Required	NO
Data Type	Attribute – xs:boolean
Data Constraints	"`true`" or "`false`" (default)
Examples	`Executable="true"`

6.18 ModuleSurfaceArea.UserExtensions

The following is the description of the `ModuleSurfaceArea.UserExtensions` instance:

```
<UserExtensions
    UserId=" xs:NCName " {1}
    Identifier=" xs:string " {1}
    AnyAttribute {0,} >
     Mixed Content, any user defined elements are also permitted.
     Any Text or XML Format
</UserExtensions>
```

Table 297. ModuleSurfaceArea.UserExtensions

Description	This section is used for any processing instructions that may be custom to the content provided by the distribution that are common to module. The content is vendor specific. The content can be plain text as well as any user-defined, properly formatted XML structure
Required	NO
Data Type	Element – Complex
Data Constraints	None
Examples	N/A

6.18.1 UserExtensions:UserId

Table 298. UserExtensions:UserId

Description	The normative reference name to identify the originator of this information.
Required	YES
Data Type	Attribute – xs:NCName
Data Constraints	A string that starts with a letter followed by any combination of letters, digits, underscores, and periods. No whitespace characters are allowed.
Examples	**Name=”NoSuchCorp”** **Name=”FooBarMfg”**

6.18.2 UserExtensions:Identifier

Table 299. UserExtensions:Identifier

Description	The string reference identify this information.
Required	YES
Data Type	Attribute– xs:normalizedString
Data Constraints	A string that starts with a number or a letter followed by any combination of letters, digits, underscores, dashes and periods. Whitespace characters are allowed.
Examples	**Name="Special Build Rules 1"** **Name="f6665cf5-8290-4b02-ba0c-5cb5a9542176"**

7
Tools Surface Area Description

This section is used for distributing various documentation, configuration templates and tools.

The XML Instance Representation for **DistributionPackage.Tools** is as follows:

```
<Tools>
    <Header> ... </Header> {0,1}
    <Filename> ... </Filename> {1,}
</Tools>
```

Table 300. DistributionPackage.Tools

Description	This element contains all the information associated with a Tools Surface Area Description
Required	NO
Data Type	Element – Complex
Data Constraints	The Header element is required. All other elements are optional.
Examples	N/A

7.1 Tools.Header

The following is the description of the **Tools.Header** instance:

```
<Header>
  <Name> xs:normalizedString </Name> {1}
  <Copyright> xs:string </Copyright> {0,1}
  <License> xs:string </License> {0,1}
  <Abstract> xs:normalizedString </Abstract> {0,1}
  <Description> xs:string </Description> {0,1}
</Header>
```

Table 301. Tools.Header

Description	This element contains the header information for a Tools Surface Area Description. This includes the name of the tools package, descriptions of the tools package, copyright and licensing information associated with the tools package, and the version of XML Schema to which this file conforms.
Required	NO
Data Type	Element – Complex
Data Constraints	N/A
Examples	N/A

7.1.1 Header.Name

Table 302. Header.Name

Description	The User Interface Name for the Tools package.
Required	YES
Data Type	Element – xs:normalizedString
Data Constraints	String value of more than one character.
Examples	`<Name>BaseTools Package, Version 2.0</Name>` `<Name>AntTools Package, Version 1.2</Name>`

7.1.2 Header.Copyright

Table 303. Header.Copyright

Description	The copyright for this tools package if it is different than the copyright of the distribution package. The copyright is generated by the creator of a package. If a derivative work is generated from an existing package, then the existing copyright must be maintained, and additional copyrights may be appended to the end of this element.
Required	NO
Data Type	Element – xs:string
Data Constraints	A set of one or more copyright statements on one or more lines of text.
Examples	`<Copyright> Copyright (c) 2008, Nosuch Corporation. All rights reserved. </Copyright>`

7.1.3 Header.License

Table 304. Header.License

Description	A license that describes any restrictions on the use of this tools package. If a derivative work is allowed by the original license and a derivative work is generated from an existing package, then the existing license must be maintained, and additional licenses may be appended to the end of this element.

Required	NO
Data Type	Element – xs:string
Data Constraints	A string that contains one or more lines of text.
Examples	**<License>** **This program and the accompanying materials are licensed and made available under the terms and conditions of the BSD License which accompanies this distribution. The full text of the license may be found at** **http://opensource.org/licenses/bsd-license.php** **THE PROGRAM IS DISTRIBUTED UNDER THE BSD LICENSE ON AN "AS IS" BASIS, WITHOUT WARRANTIES OR REPRESENTATIONS OF ANY KIND, EITHER EXPRESS OR IMPLIED.** **</License>**

7.1.4 Header.Abstract

Table 305. Header.Abstract

Description	A brief text description of the tools. This description must include the release name of the tools, the version of the tools, and a description of the tools contents and/or features.
Required	NO
Data Type	Element – xs:normalizedString
Data Constraints	A string that contains two or more words.
Examples	`<Abstract> Win32 Binaries and template files for Makefile Base Build system </Abstract>`

7.1.5 Header.Description

Table 306. Header.Description

Description	A complete description of a tools package. This description should include the release name of the package, the version of the package, and a complete description of the package contents and/or features including a description of the updates since the previous package release.
Required	NO
Data Type	Element – xs:string
Data Constraints	A string that contains one or more lines of text.
Examples	`<Description> BaseTools package using Makefile based builds. The binaries provided are Win32 executables, which can execute on both IA32 and X64 based operating systems. UNIX based operating systems will need to compile the C tools which are POSIX compliant. Python tools may be compiled to executable format, or run under the Python interpreter. </Description>`

7.2 Tools.Filename

This section is used for distributing both source and binary files for tools. If the Exec bit is not set, then the file is either a source file, a configuration file, or written documentation.

The following is the description of the **`Tools.Filename`** instance:

```
<Filename
  OS=" SupportedOs " {0,1}
  Executable=" xs:boolean " {0,1} >
     xs:anyURI
</Filename> {1,}
```

Table 307. Tools.Filename

Description	Any description, executable, template or configuration file that make up the infrastructure and build system.
Required	YES
Data Type	Element – xs:anyURI
Data Constraints	The path and filename of the file as it appears in the Distribution Content FileFile.
Examples	**<Filename Root="true">BuildNotes2.txt</Filename>** **<Filename>Conf/tools_def.template</Filename>** **<Filename Root="true">edksetup.bat</Filename>** **<Filename Revision="937">Bin/Win32/build.exe</Filename>**

7.2.1 Filename:OS

Table 308. Filename:OS

Description	Used to identify a specific operating system this tool is targeted for. This is informational only.
Required	NO
Data Type	Attribute – xs:NCName
Data Constraints	Enumeration: **Win32**, **Win64**, **Linux32**, **Linux64**, **OS/X32**, **OS/X64**, **GenericWin**, **GenericNix** or any single word identifier.
Examples	**OS="Win32"**

7.2.2 Filename:Executable

Table 309. Filename:Executable

Description	This flag is used during installation to ensure that the file is executable.
Required	NO
Data Type	Attribute – xs:boolean
Data Constraints	"**true**" or "**false**" (default – not present)
Examples	**Executable="true"**

8
MiscellaneousFiles Section

This section describes the miscellaneous files section of the Distribution Description. These sections are used to list files that do not fall into previous categories. The distribution package provider must provide information as to how this information will be used.

The XML Instance Representation for **DistributionPackage.MiscellaneousFiles** is as follows:

```
<MiscellaneousFiles
    <Header> ... </Header> {0,1}
    <Filename> ... </Filename> {0,}
</MiscellaneousFiles>
```

Table 310. DistributionPackage.MiscellaneousFiles

Description	This element contains any miscellaneous distribution information and/or files not defined previously. Processing of this section is specified as “lax”.
Required	NO
Data Type	Element – Complex
Data Constraints	Free-form text or any well formed XML element(s.)
Examples	N/A

8.1 MiscellaneousFiles.Header

The following is the description of the **MiscellaneousFiles.Header** instance:

```
<Header>
  <Name> xs:normalizedString </Name> {0,1}
  <Copyright> xs:string </Copyright> {0,1}
  <License> xs:string </License> {0,1}
  <Abstract> xs:normalizedString </Abstract> {0,1}
  <Description> xs:string </Description> {0,1}
</Header>
```

Table 311. MiscellaneousFiles.Header

Description	This element contains the header information for a Misc Section.
Required	NO
Data Type	Element – Complex
Data Constraints	N/A
Examples	N/A

8.1.1 Header.Name

Table 312. Header.Name

Description	The User Interface Name for the Miscellaneous content.
Required	YES
Data Type	Element – xs:normalizedString
Data Constraints	String value of more than one character.
Examples	`<Name>BaseTools Package, Version 2.0</Name>` `<Name>AntTools Package, Version 1.2</Name>`

8.1.2 Header.Copyright

Table 313. Header.Copyright

Description	The copyright for this miscellaneous content if it is different than the copyright of the distribution package. The copyright is generated by the creator of the content. If a derivative work is generated from an existing work, then the existing copyright must be maintained, and additional copyrights may be appended to the end of this element.
Required	NO
Data Type	Element – xs:string
Data Constraints	A set of one or more copyright statements on one or more lines of text.

Examples	`<Copyright>`
	`Copyright (c) 2008, Nosuch Corporation. All rights reserved.`
	`</Copyright>`

8.1.3 Header.License

Table 314. Header.License

Description	A license that describes any restrictions on the use of this content. If a derivative work is allowed by the original license and a derivative work is generated from an existing package, then the existing license must be maintained, and additional licenses may be appended to the end of this element.

Required	NO
Data Type	Element – xs:string
Data Constraints	A string that contains one or more lines of text.
Examples	<License> This program and the accompanying materials are licensed and made available under the terms and conditions of the BSD License which accompanies this distribution. The full text of the license may be found at http://opensource.org/licenses/bsd-license.php THE PROGRAM IS DISTRIBUTED UNDER THE BSD LICENSE ON AN "AS IS" BASIS, WITHOUT WARRANTIES OR REPRESENTATIONS OF ANY KIND, EITHER EXPRESS OR IMPLIED. </License>

8.1.4 Header.Abstract

Table 315. Header.Abstract

Description	A brief text description of the content.
Required	NO
Data Type	Element – xs:normalizedString
Data Constraints	A string that contains two or more words.
Examples	<Abstract> Applicable Specifications </Abstract>

8.1.5 Header.Description

Table 316. Header.Description

Description	A complete description of the content.
Required	NO
Data Type	Element – xs:string
Data Constraints	A string that contains one or more lines of text.
Examples	<Description> Specification PDF documents ACPI WG. Dated 2004. </Description>

8.2 MiscellaneousFiles.Filename

The following is the description of the **MiscellaneousFiles.Filename** instance:

```
<Filename
  Executable=" xs:boolean " {0,1} >
    xs:anyURI
</Filename>
```

Table 317. MiscellaneousFiles.Filename

Description	A file that is included in the distribution package that does not pertain to any of the previously defined sections.
Required	NO
Data Type	Element – xs:anyURI
Data Constraints	The path and filename of the file as it appears in the Distribution Content File.
Examples	`<File>FooBarMfgBuildReadMe.txt</File>`

8.2.1 Filename:Executable

Table 318. Filename:Executable

Description	This flag is used during installation to ensure that the file is executable.
Required	NO
Data Type	Attribute – xs:boolean
Data Constraints	"`true`" or "`false`" (default)
Examples	`Executable="true"`

9
UserExtensions

This section describes the UserExtensions section of the Distribution Description File. These sections are used to describe any extensions to this specification. The distribution package provider must provide information as to how this information will be used.

The following is a description of the **`DistributionPackage.UserExtensions`** instance:

```
<UserExtensions
  UserId=" xs:NCName " {1} >
  Identifier=" xs:string " {0,1}
  AnyAttribute {0,} >
    mixed="true"
    Any Content is permitted
</UserExtensions>
```

Table 319. DistributionPackage.UserExtensions

Description	This element contains any miscellaneous distribution information not defined previously. Processing of this section is specified as “lax”.
Required	NO
Data Type	Element – Complex, mixed=true
Data Constraints	None
Examples	N/A

9.1 UserExtensions:UserId

Table 320. UserExtensions:UserId

Description	The normative reference name to identify the originator of this information.
Required	NO
Data Type	Attribute– xs:NCName
Data Constraints	A single word identifier for grouping similar content.
Examples	`UserId=”NoSuchCorp”`

9.2 UserExtensions:Identifier

Table 321. UserExtensions:Identifier

Description	A single string that can be used to sub-divide content of a group of items.
Required	NO
Data Type	Attribute– xs:string
Data Constraints	A string identifier for sub-grouping content.
Examples	`Identifier=”PRE_BUILD”` `Identifier="f6665cf5-8290-4b02-ba0c-5cb5a9542176"`

Appendix A
XML Examples

This section contains several XML examples. These examples are either portions of a distribution package or simple examples of a distribution package containing a single component. Typical use cases will likely be more complex than shown in this appendix because distribution packages will likely contain several components of each type. The following XML example are included in this section:

- Distribution Package Header
- Package Header
- Distribution Package containing source to a UEFI Application
- Distribution Package containing binary for a UEFI Drivers
- Distribution Package containing source for a Library
- Distribution Package containing source for a PI PEIM
- Distribution Package containing source for a DXE Driver
- Distribution Package containing a binary of a Tool

A.1 Distribution Package Header

The following example is the header from a distribution package for chipset related content. This distribution package is read-only and consumers of this package are not allowed to repackage this content into another package. This distribution package is from Nosuch Corporation, and contains the base data types and libraries for implementing chipset modules for Si products produced by Nosuch Corporation. The version of the release is 1.2 and it was created on 3/24/2008 at 09:30. It is covered by a BSD license and the Nosuch Corporation owns the copyright for the content.

```
<DistributionHeader  ReadOnly="true" RePackage="false">
  <Name BaseName="NosuchChipset">Nosuch Chipset</Name>
  <GUID Version="1.2"> AF0DDA2E-EA83-480b-B2CE-FC0BB2F894C2 </
GUID>
  <Vendor> Nosuch Corporation </Vendor>
  <Date>2008-03-24T09:30:00</Date>
  <Copyright>Copyright (c)  2008, Nosuch Corporation. All rights
reserved.</Copyright>
  <License>
    This program and the accompanying materials are licensed and
made available under
    the terms and conditions of the BSD License which accompanies
this distribution.
    The full text of the license may be found at:
    http://opensource.org/licenses/bsd-license.php
    THE PROGRAM IS DISTRIBUTED UNDER THE BSD LICENSE ON AN "AS
IS" BASIS, WITHOUT
    WARRANTIES OR REPRESENTATIONS OF ANY KIND, EITHER EXPRESS OR
IMPLIED.
  </License>
  <Abstract> Version 1.2 of the NoSuchChipset Distribution
Package. </Abstract>
  <Description>
    Initial release of the NoSuchChipset, Version 1.0 providing
the base data types
    and libraries for NoSuch UEFI Driver. This includes special
libraries for
    reporting status codes of the driver.
  </Description>
  <Signature> 09fa8fc7222da9afd9ffd52ba8b73f45 </Signature>
  <XmlSpecification> 1.0 </XmlSpecification>
</DistributionHeader>
```

A.2 Package Header

The following example is the package surface area for the UEFI or PI Packages that provide the base types and defintions from the *UEFI 2.2 Specification* and *PI 1.2 Specification*. It is implemented by Nosuch Corporation, covered by a copyright from Nosuch Corporation, and licensed under BSD. This abbreviated example shows the public include files **`Uefi.h`**, **`PiPeim.h`**, and **`PiDxe.h`** along with the GuidValue for EFI global variables, the Protocol GUID for the EFI Block I/O Protocol that is only available to a subset of the DXE and UEFI module types, and a token space GUID for Platform Configuration Database (PCD) tokens defined by Nosuch Corporation. It also declares an I/O Port Library class and a PCD that are not related to the UEFI or PI Specification content, but are listed here because they are referenced by additional examples later in this section.

```
<PackageSurfaceArea>
      <Header>
    <Name BaseName="UefiPiPkg"> UEFI/PI Package </Name>
    <GUID Version="1.0"> 26E136A4-1060-48fb-AEB6-3DB66C175CFB </
GUID>
    <Copyright>
      Copyright (c) 2008, Nosuch Corporation. All rights
reserved.
    </Copyright>
    <License>
      This program and the accompanying materials are licensed
and made available
      under the terms and conditions of the BSD License which
accompanies this
      distribution.  The full text of the license may be found
at:
      http://opensource.org/licenses/bsd-license.php
      THE PROGRAM IS DISTRIBUTED UNDER THE BSD LICENSE ON AN "AS
IS" BASIS, WITHOUT
      WARRANTIES OR REPRESENTATIONS OF ANY KIND, EITHER EXPRESS
OR IMPLIED.
    </License>
    <Abstract> Version 1.0 of the UEFI/PI Package. </Abstract>
    <Description>
      Provides the base data types and definitions from the UEFI
2.2 Specification
      and the PI 1.2 Specification
    </Description>
    <PackagePath> UefiPiPkg </PackagePath>
</Header>
  <LibraryClassDeclarations>
    <LibraryClass Keyword="IoPortLib">
      <HeaderFile> Include/Library/IoPortLib.h </HeaderFile>
    </LibraryClass>
  </LibraryClassDeclarations>
  <PackageIncludes>
    <PackageHeader>
      <HeaderFile> Include/Uefi.h </HeaderFile>
    </PackageHeader>
    <PackageHeader>
      <HeaderFile> Include/PiPeim.h </HeaderFile>
    </PackageHeader>
    <PackageHeader>
      <HeaderFile> Include/PiDxe.h </HeaderFile>
    </PackageHeader>
</PackageIncludes>
  <GuidDeclarations>
```

```
    <Entry UiName="EFI Global Variable GUID"
GuidTypes="Variable">
      <CName> gEfiGlobalVariableGuid </CName>
      <GuidValue> 8BE4DF61-93CA-11D2-AA0D-00E098032B8C </
GuidValue>
  </Entry>
    <Entry UiName="Toke Space GUID for Nosuch Corp PCDs"
GuidTypes="TokenSpaceGuid">
      <CName> gNosuchCorpTokenSpaceGuid </CName>
      <GuidValue> 5702D1FD-AD66-4c14-980F-DF9F34113B81 </
GuidValue>
  </Entry>
</GuidDeclarations>
<ProtocolDeclarations>
   <Entry UiName="Block I/O Protocol" SupModuleList="DXE_DRIVER
DXE_RUNTIME_DRIVER UEFI_DRIVER UEFI_APPLICATION">
      <CName> gEfiBlockIoProtocolGuid </CName>
      <GuidValue> 964E5B21-6459-11D2-8E39-00A0C969723B </
GuidValue>
         <HelpText>
       The Block I/O Protocol provides services to read, write,
and flush block to a
        block oriented storage device such as a hard disk, CD-
ROM, DVD, and floppy.
        This Protocol is available to all CPU types, but is
restricted for use by
        DXE_DRIVER, UEFI_RUNTIME_DRIVER, UEFI_DRIVER, and
UEFI_APPLICATION module
        types.
      </HelpText>
    </Entry>
</ProtocolDeclarations>
  <PcdDeclarations>
    <PcdEntry>
      <TokenSpaceGuidCname> gNosuchCorpTokenSpaceGuid </
TokenSpaceGuidCname>
      <Token> 00000001 </Token>
      <CName> SmBusHostControllerIoPortBaseAddress </CName>
      <DatumType> UINT32 </DatumType> {1}
      <ValidUsage> FixedPcd PatchPcd Pcd PcdEx </ValidUsage>
      <DefaultValue> 0x500 </DefaultValue>
    </PcdEntry>
  </PcdDeclarations>
</PackageSurfaceArea>
```

A.3 UEFI Application

The following example is the distribution package that contains a package, and that package contains the source code to a module that is an EFI Application that displays the message “Hello World” on the console output devices. All of the material in this distribution package are implemented by Nosuch Corporation, covered by a copyright from Nosuch Corporation, and licensed under BSD. The EFI application depends on the UefiPkg described in Section A.2 so the application can use `Uefi.h` that contains the base types and defintions from the *UEFI 2.2 Specification*.

```
<?xml version="1.0"?>
<DistributionPackage xmlns="http://www.uefi.org/2008/2.1"
xmlns:xsi="http:/www.w3.org/2001/XMLSchema-instance" >
  <DistributionHeader  ReadOnly="true" RePackage="false">
    <Name BaseName="HelloWorldDistribuition">Hello World
Distribution</Name>
    <GUID Version="1.0"> 62D1DDD0-FC9C-4809-BE6D-DFB8B463B1D6 </
GUID>
    <Vendor> Nosuch Corporation </Vendor>
    <Date>2008-03-24T09:30:00</Date>
    <Copyright>
      Copyright (c)  2008, Nosuch Corporation. All rights
reserved.
    </Copyright>
    <Abstract> Version 1.0 of the Hello World Distribution
Package. </Abstract>
    <XmlSpecification> 1.0 </XmlSpecification>
  </DistributionHeader>
  <PackageSurfaceArea>
    <Header>
      <Name BaseName="HelloWorldPkg"> Hello World Package </Name>
      <GUID Version="1.0"> 0410A3C9-6875-4010-8EC1-0BC1FD96D9AC
</GUID>
      <Copyright>
        Copyright (c) 2008, Nosuch Corporation. All rights
reserved.
      </Copyright>
      <Abstract> Version 1.0 of the Hello World Package. </
Abstract>
      <PackagePath> HelloWorldPkg </PackagePath>
  </Header>
    <ModuleSurfaceArea  BinaryModule="false">
      <Header>
        <Name BaseName="HelloWorld"> Hello World </Name>
       <GUID Version="1.0"> 179F65BD-0092-455b-81A5-09B1CD27EA33
</GUID>
        <Copyright>
          Copyright (c) 2008, Nosuch Corporation. All rights
reserved.
        </Copyright>
        <Abstract> Hello World UEFI Application </Abstract>
        <Description>
          Displays the message 'Hello World' on all UEFI console
output devices
        </Description>
      </Header>
      <ModuleProperties>
```

```
        <ModuleType> UEFI_APPLICATION </ModuleType>
        <Path> HelloWorld </Path>
        <UefiSpecificationVersion> 2.0 </
UefiSpecificationVersion>
      </ModuleProperties>
      <SourceFiles>
        <Filename> HelloWorld.c </Filename>
      </SourceFiles>
      <PackageDependencies>
        <Package>
          <GUID Version="1.0"> 26E136A4-1060-48fb-AEB6-
3DB66C175CFB</GUID>
        </Package>
      </PackageDependencies>
      <Externs>
        <Extern>
          <EntryPoint> InitializeHelloWorld </EntryPoint>
        </Extern>
      </Externs>
    </ModuleSurfaceArea>
  </PackageSurfaceArea>
</DistributionPackage>
```

A.4 UEFI Application Source Files

```
#include <Uefi.h>

EFI_STATUS
EFIAPI
InitializeHelloWorld (
  IN EFI_HANDLE        ImageHandle,
  IN EFI_SYSTEM_TABLE  *SystemTable
  )
{
  SystemTable->ConOut->OutputString (SystemTable->ConOut,
L"Hello World\n");
  return EFI_SUCCESS;
}
```

A.5 UEFI Driver

This examples contains the Module Surface Area for a UEFI Driver binary for a PCI graphics controller from Nosuch Corporation. Binary images for IA32, X64, IPF, and EBC are included. The driver follows the UEFI Driver Model and produces the EFI Driver Bindng Protocol. This driver uses the services of the EFI PCI I/O Protocol to produce the Graphics Ouput Protocol services.

```
    <ModuleSurfaceArea  BinaryModule="true">
      <Header>
       <Name BaseName="NosuchUefiGop"> Nosuch UEFI GOP Driver </
Name>
       <GUID Version="1.0"> F32C4FAC-F5E9-4cdc-B8C9-48AC276CB87F
</GUID>
        <Copyright>
          Copyright (c) 2008, Nosuch Corporation. All rights
reserved.
        </Copyright>
        <Abstract> UEFI Graphics Output Protocol driver </
Abstract>
        <Description>
          Produces the Graphics Output Protocol for PCI graphics
controllers from
           Nosuch Corporation
        </Description>
      </Header>
      <ModuleProperties>
        <ModuleType> UEFI_DRIVER </ModuleType>
        <Path> NosuchUefiGop </Path>
        <UefiSpecificationVersion> 2.0 </
UefiSpecificationVersion>
      </ModuleProperties>
      <BinaryFiles>
        <BinaryFile>
          <Filename FileType="PE32" SupArchList="IA32">
            Ia32/NosuchUefiGop.efi
          </Filename>
          <Filename FileType="PE32" SupArchList="X64">
            X64/NosuchUefiGop.efi
          </Filename>
          <Filename FileType="PE32" SupArchList="IPF">
            Ipf/NosuchUefiGop.efi
          </Filename>
          <Filename FileType="PE32" SupArchList="EBC">
            Ebc/NosuchUefiGop.efi
          </Filename>
        </BinaryFile>
      </BinaryFiles>
      <Protocols>
        <Protocol Usage="PRODUCES">
          <CName> gEfiDriverBindingProtocolGuid </CName>
        </Protocol>
        <Protocol Usage="TO_START">
          <CName> gEfiPciIoProtocolGuid </CName>
        </Protocol>
```

```
      <Protocol Usage="BY_START">
        <CName> gEfiGraphicsOutputProtocolGuid </CName>
      </Protocol>
    </Protocols>
  </ModuleSurfaceArea>
```

A.6 Library

This example contains the Module Surface Area for the source code to a library from Nosuch Corpration that produces the a set APIs to access I/O ports on IA32 and X64 platforms. On IA32 and X64 platforms, this is implemented using IN and OUT instructions. In this implementation assembly code is required to perform IN and OUT instructions. Both the MSFT and GCC versions of the ASM code for both IA32 and X64 are provided so this library can be built using a number of different tool chains.

```
    <ModuleSurfaceArea  BinaryModule="false">
      <Header>
        <Name BaseName="IoPortLib">
          I/O Port Library
        </Name>
       <GUID Version="1.0"> E9491CAD-9457-4fac-A856-87EC9DB60A23
</GUID>
        <Copyright>
          Copyright (c) 2008, Nosuch Corporation. All rights
reserved.
        </Copyright>
        <Abstract>
          This library provides APIs to access I/O ports on IA32
and X64 platforms.
        </Abstract>
      </Header>
      <ModuleProperties SupArchList="IA32 X64">
        <ModuleType> BASE </ModuleType>
        <Path> IoPortLib </Path>
      </ModuleProperties>
      <LibraryClassDefinitions>
        <LibraryClass Usage="PRODUCES" >
          <Keyword> IoPortLib </Keyword>
        </LibraryClass>
      </LibraryClassDefinitions>
      <SourceFiles>
        <Filename> IoPortLib.c </Filename>
        <Filename Family="MSFT" SupArchList="IA32"> Ia32/
InOut.asm </Filename>
       <Filename Family="MSFT" SupArchList="X64"> X64/InOut.asm
</Filename>
        <Filename Family="GCC" SupArchList="IA32"> Ia32/InOut.S
</Filename>
        <Filename Family="GCC" SupArchList="X64"> X64/InOut.S </
Filename>
      </SourceFiles>
      <PackageDependencies>
        <Package>
          <GUID Version="1.0"> 26E136A4-1060-48fb-AEB6-
3DB66C175CFB</GUID>
        </Package>
      </PackageDependencies>
      <PPIs>
        <Ppi Usage="PRODUCES">
          <CName> gEfiPciCfg2PpiGuid </CName>
        </Ppi>
      </PPIs>
```

```
    <Externs>
      <Extern>
        <EntryPoint> PeimInitializePciCfg2 </EntryPoint>
      </Extern>
    </Externs>
  </ModuleSurfaceArea>
```

A.7 PI PEIM

This examples contains the Module Surface Area for the sources to a PEIM that is implemented by Nosuch Corporation and produces the PEI CFG2 PPI using the 0xCF8 and 0xCFC I/O ports to generate PCI configuration cycles. The dependency expression for this PEIM is TRUE. This PEIM uses an I/O Port Library to access the 0xCF8 and 0xCFC I/O ports, and has a module entry point called `PeimInitializePciCfg2`.

```
    <ModuleSurfaceArea  BinaryModule="false">
      <Header>
        <Name BaseName="PciCfg2Pei">
          PCI Configuration2 PEIM
        </Name>
       <GUID Version="1.0"> EA3AA39C-2AC1-44ef-8D2D-D57E34C9FD4C
</GUID>
        <Copyright>
          Copyright (c) 2008, Nosuch Corporation. All rights
reserved.
        </Copyright>
        <Abstract>
          Produces the PCI CFG2 PPI using I/O ports 0xCF8 and
0xCFC to perform
          PCI configuration cycles.
        </Abstract>
      </Header>
      <ModuleProperties>
        <ModuleType> PEIM </ModuleType>
        <Path> PciCfg2Pei </Path>
        <PiSpecificationVersion> 1.0 </PiSpecificationVersion>
      </ModuleProperties>
      <LibraryClassDefinitions>
        <LibraryClass Usage="CONSUMES" >
          <Keyword> IoPortLib </Keyword>
        </LibraryClass>
      </LibraryClassDefinitions>
      <SourceFiles>
        <Filename> PciCfg2.c </Filename>
      </SourceFiles>
      <PackageDependencies>
        <Package>
          <GUID Version="1.0"> 26E136A4-1060-48fb-AEB6-
3DB66C175CFB</GUID>
        </Package>
      </PackageDependencies>
      <PPIs>
        <Ppi Usage="PRODUCES">
          <CName> gEfiPciCfg2PpiGuid </CName>
        </Ppi>
      </PPIs>
      <Externs>
        <Extern>
          <EntryPoint> PeimInitializePciCfg2 </EntryPoint>
        </Extern>
      </Externs>
      <PeiDepex>
```

```
      <Expression> TRUE </Expression>
    </PeiDepex>
  </ModuleSurfaceArea>
```

A.8 PI DXE Driver

This examples contains the Module Surface Area for the sources to a DXE Driver that is implemented by Nosuch Corporation and produces the SMBUS Host Controller Protrocol using I/O ports to access the SMBUS host controller hardware. The dependency expression for this DXE Driver is the Metronome Architectural Protocol because some of the SMBUS Host Controller Protocol services require a timeout. This DXE Driver uses the I/O Port Library to access the I/O ports for the SMBUS host controller, and the base I/O port address for the SMBUS Host Controller is defined by a the Platform Configuration Database token called **`SmBusHostControllerIoPortBaseAddress`**. The entry point to this module is called **`InitializeSmBusHostController`**.

```
    <ModuleSurfaceArea  BinaryModule="false">
      <Header>
        <Name BaseName="SmBusHostControllerDxe ">
          SMBUS Host Controller DXE Driver
        </Name>
       <GUID Version="1.0"> EF514376-72C0-43e3-B4C7-2AE8D4D0848A
</GUID>
        <Copyright>
          Copyright (c) 2008, Nosuch Corporation. All rights
reserved.
        </Copyright>
        <Abstract>
          Produces the SMBUS Host Controller Protocol in the
Nosuch Corporation
         chipset.  The SMBUS controller in this chipset is access
through I/O
          ports whose I/O port base address is programmable.  The
I/O port base
          is specified by the Platform Configuration Database
token
          SmBusHostControllerIoPortBaseAddress.
        </Abstract>
      </Header>
      <ModuleProperties>
        <ModuleType> DXE_DRIVER </ModuleType>
        <Path> SmBusHostControllerDxe </Path>
        <PiSpecificationVersion> 1.0 </PiSpecificationVersion>
      </ModuleProperties>
      <LibraryClassDefinitions>
        <LibraryClass Usage="CONSUMES" >
          <Keyword> IoPortLib </Keyword>
        </LibraryClass>
      </LibraryClassDefinitions>
      <SourceFiles>
        <Filename> SmBusHostController.c </Filename>
      </SourceFiles>
      <PackageDependencies>
        <Package>
          <GUID Version="1.0"> 26E136A4-1060-48fb-AEB6-
3DB66C175CFB</GUID>
        </Package>
      </PackageDependencies>
      <Protocols>
        <Protocol Usage="PRODUCES">
          <CName> gEfiSmbusHostControllerProtocolGuid </CName>
        </Protocol>
        <Protocol Usage="CONSUMES">
```

```
          <CName> gEfiMetronomeArchProtocolGuid </CName>
        </Protocol>
      </Protocols>
      <Externs>
        <Extern>
          <EntryPoint> InitializeSmBusHostController </
EntryPoint>
        </Extern>
      </Externs>
      <PcdCoded>
        <PcdEntry PcdItemType="Pcd" PcdUsage="CONSUMES">
          <CName> SmBusHostControllerIoPortBaseAddress </CName>
          <TokenSpaceGuidCName> gNosuchCorpTokenSpaceGuid </
TokenSpaceGuidCName>
        </PcdEntry>
      </PcdCoded>
      <DxeDepex>
        <Expression> gEfiMetronomeArchProtocolGuid </Expression>
      </DxeDepex>
    </ModuleSurfaceArea>
```

A.9 Tool

This examples contains an executable utility from Nosuch Corporation that help print UEFI source files. Two versions of the tool are included. One for Win32 and the other for Win64.

```
<Tools>
  <Header>
    <Name> UEFI Print Utility Version 1.0 </Name>
    <Copyright>
      Copyright (c) 2008, Nosuch Corporation. All rights
reserved.
    </Copyright>
    <Abstract> Utility to print UEFI source files </Abstract>
  </Header>
<Filename OS="Win32" Executable="true">Win32/UefiPrint.exe</
Filename>
<Filename OS="Win64" Executable="true">Win64/UefiPrint.exe</
Filename>
</Tools>
```