



# VFR Programming Language

*Feb 12, 2010*

*Revision 1.3*

THIS SPECIFICATION IS PROVIDED "AS IS" WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NONINFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION OR SAMPLE. Except for a limited copyright license to copy this specification for internal use only, no license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted herein.

Intel disclaims all liability, including liability for infringement of any proprietary rights, relating to implementation of information in this specification. Intel does not warrant or represent that such implementation(s) will not infringe such rights.

Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them.

This document is an intermediate draft for comment only and is subject to change without notice. Readers should not design products based on this document.

Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

\*Other names and brands may be claimed as the property of others.

Copyright © 2007, 2008, 2009, 2010 Intel Corporation. All rights reserved.

# Contents

---

1	Introduction .....	1
1.1	Overview .....	1
1.2	Target Audience .....	1
1.3	Related Information.....	1
1.4	Terms.....	1
2	VFR Description in BNF .....	3
2.1	VFR Programming Keywords .....	3
2.1.1	// (comment marker).....	3
2.1.2	#define .....	4
2.1.3	#include .....	4
2.2	VFR Program .....	4
2.3	VFR Data Struct Definition .....	5
2.4	VFR FormSet Definition .....	6
2.5	VFR FormSet List Definition .....	7
2.6	VFR Default Stores Definition .....	7
2.7	VFR Variable Store Definition .....	8
2.7.1	VFR Buffer Store Definition .....	8
2.7.2	VFR EFI Variable Store Definition.....	8
2.7.3	VFR Variable Name Store Definition .....	8
2.8	VFR FormSet DisableIf Definition .....	9
2.9	VFR FormSet SuppressIf Definition.....	9
2.10	VFR General Token Definition.....	10
2.10.1	GUID Definition.....	10
2.10.2	String & String Identifier Definition .....	10
2.10.3	Number Definition .....	10
2.10.4	VFR Statement Header Definition.....	10
2.10.5	VFR Question Header Definition.....	11
2.10.6	VFR Constant Value Definition .....	11
2.10.7	VFR Statement Image & Locked Tag Definition .....	12
2.11	VFR Form Definition.....	12
2.11.1	VFR Form Map Definition .....	13
2.11.2	VFR Image Statement Definition.....	13
2.11.3	VFR Locked Statement Definition.....	13
2.11.4	VFR Rule Statement Definition .....	13
2.11.5	VFR Statement Definition .....	14
2.11.6	VFR Question Type Statements Definition .....	17
2.11.7	VFR Conditional Type Statements Definition .....	32
2.11.8	VFR GUID Statement Definition .....	33
2.12	VFR Expression Statement Definition.....	34
2.12.1	OR .....	34
2.12.2	AND .....	34
2.12.3	bitwiseor .....	34
2.12.4	bitwiseand .....	34
2.12.5	equal.....	35
2.12.6	compare .....	35
2.12.7	shift .....	35

2.12.8	add/minus.....	35
2.12.9	multiply/divide/modulo .....	36
2.12.10	cast terms.....	36
2.12.11	atom terms .....	36

## *Revision History*

---

Revision Number	Description	Revision Date
0.9	Preliminary release	December 12, 2007
1.0	First general release	December 21, 2007
1.1	Update some syntax and fix some literal issues	February 20, 2008
1.2	Add more defaultstore ID description and fix some literal errors	May 26, 2008
1.3	Add new syntax classguid, suppressiformset, security, formmap to support UEFI2.3 HII IFR opcodes, update some syntax and fix some literal errors.	Feb 12, 2010



# 1

## *Introduction*

---

### 1.1 Overview

Internal Forms Representation (IFR) is a binary encoding. This encoding is designed to be comparatively easy to parse and manipulate programmatically.

To simplify the creation of IFR, a high-level Visual Forms Representation (VFR) language is defined below. Using this language syntax, a compiler can be designed which would take an ordinary text file containing VFR as an input, and emit IFR that could be used in a user's program. There can be a variety of methods to define the VFR language. In this document, we send out a method and describe it with a BNF-style syntax.

### 1.2 Target Audience

This document is for the readers who will adopt the VFR language in designing and developing their products with UEFI 2.3 compliance. It may be used as reference manual by developers. It offers enough material to create infrastructure files for the user interface as well as the creation of a driver to export setup related information to the Human Interface Infrastructure programming interface.

A full understanding of the UEFI 2.3 Specification is assumed throughout this document.

### 1.3 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 2.3, Unified EFI, Inc,  
<http://www.uefi.org>

### 1.4 Terms

The following terms are used throughout this document to describe varying aspects of input localization:

#### **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.

**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.

**VFR**

Visual Forms Representation. This is the source code format that is used by developers to create a user interface with varying pieces of data or questions. This is later compiled into a binary encoding



## 2

# VFR Description in BNF

---

This section describes an example development language that developers should use to write forms. This language is compiled into an architectural form (i.e. IFR) described in the UEFI 2.3 specification. Each section in the chapter describes a different language terminal or non-terminal.

The VFR language uses a BNF-style syntax:

- For the BNF-style syntax used here, literal characters and terms are in bold and marked with quotation mark.
- Terms are case-sensitive. VFR statements start with `“//”` and end at the end of the line.
- Terms enclosed in non-bolded braces `{ }` are optional.
- Terms enclosed in parenthesis `( )` and followed by an asterisk may be specified in the input VFR zero or more times.
- If the parenthesis is followed by a plus `“+”` sign, then the term must be present one or more times.
- Groups of terms are sometimes enclosed in parenthesis, and the character `“|”` between the two terms indicates that either term is acceptable syntax.
- A super script number, `n`, followed by a comma `“,”` and another number, `m`, such as `item(n, m)` is used to indicate that the number of occurrences can be from `n` to `m` occurrences of the item, inclusive.

**Note:** *The text formatting of the VFR statements and examples in this document are consistent with BNF-style conventions rather than with EFI or UEFI conventions.*

VFR forms consist of a series of VFR statements. The subsections in this chapter designate the base definitions of the different VFR statements by specifying the keyword `“Statement”` in the section title, followed by the BNF non-terminal name enclosed in parenthesis. Subsections without the `“Statement”` keyword in the title are non-terminal definitions referenced by one or more of the VFR statement definitions.

## 2.1 VFR Programming Keywords

The keywords described in the following section compose the working set of commands for the VFR language. An example follows each keyword description to help programmers understand how a sample line of VFR code would look.

### 2.1.1 `// (comment marker)`

Allows a programmer to leave comments in the VFR file. Has no effect on the IFR binary that is generated. Example:

```
// this is a typical comment marker
```

### 2.1.2 **#define**

This command is used to assign a meaningful name to a constant: very similar in function to the 'C' style. Example:

```
#define FORMSET_GUID \
{0xA04A27f4,0xDF00,0x4D42,0xB5,0x52,0x39,0x51,0x13,0x02,0x11,0x3D}
```

### 2.1.3 **#include**

This command tells the VFR compiler to use the contents of a file as part of the source to compile. Example:

```
#include "C:\Source\ DriverSampleStrDefs.h"
```

## 2.2 **VFR Program**

A complete VFR program takes the following form:

```
vfrProgram ::=
(
    vfrPragmaPackDefinition
    | vfrDataStructDefinition
)*
vfrFormSetDefinition

vfrPragmaPackDefinition ::=
"#pragma" "pack" "(" vfrPragmaPackType ")"

vfrPragmaPackType ::=
{
    "show"
    | { "push" | "pop" } { "," StringIdentifier } { "," Number }
    | { Number }
}
```

### **BEHAVIORS AND RESTRICTIONS:**

The data structure must be defined before formset statements. The pragma pack number must be the second power of 2. Example: None.

## 2.3 VFR Data Struct Definition

```

vfrDataStructDefinition ::=
{ "typedef" } "struct"
{ StringIdentifier }
{" vfrDataStructFields "}
{ StringIdentifier } ";"

vfrDataStructFields ::=
(
    dataStructField64
    dataStructField32
    dataStructField16
    dataStructField8
    dataStructFieldBool
    dataStructFieldString
    dataStructFieldDate
    dataStructFieldTime
    dataStructFieldUser
) *

dataStructField64 ::=
"UINT64"
StringIdentifier { "[" Number "]" } ";"

dataStructField32 ::=
"UINT32"
StringIdentifier { "[" Number "]" } ";"

dataStructField16 ::=
"UINT16"
StringIdentifier { "[" Number "]" } ";"

dataStructField8 ::=
"UINT8"
StringIdentifier { "[" Number "]" } ";"

dataStructFieldBool ::=
"BOOLEAN"
StringIdentifier { "[" Number "]" } ";"

dataStructFieldString ::=
"EFI_STRING_ID"
StringIdentifier { "[" Number "]" } ";"

dataStructFieldDate ::=
"EFI_HII_DATE"
StringIdentifier { "[" Number "]" } ";"

dataStructFieldTime ::=
"EFI_HII_TIME"
StringIdentifier { "[" Number "]" } ";"

dataStructFieldUser ::=
StringIdentifier StringIdentifier { "[" Number "]" } ";"

```

### BEHAVIORS AND RESTRICTIONS:

The data structure definition is C language style. The key word of the fields' type must be one of these types: **UINT8** **UINT16** **UINT32** **UINT64** **BOOLEAN** **EFI\_STRING\_ID** **EFI\_HII\_DATA** **EFI\_HII\_TIME** or user defined data structure. Example:

```
typedef struct {
    UINT8  mU8;
    UINT16 mU16;
    UINT32 mU32[10];
    UINT64 mU64;
} MyData;
```

## 2.4 VFR FormSet Definition

```
vfrFormSetDefinition ::=
    "formset"
    "guid" "=" guidDefinition ","
    "title" "=" getStringId ","
    "help" "=" getStringId ","
    { "classguid" "=" classguidDefinition "," }
    { "class" "=" classDefinition "," }
    { "subclass" "=" subclassDefinition "," }
    vfrFormSetList
    "endformset" ","

classguidDefinition ::=
    guidDefinition { "|" guidDefinition } { "|" guidDefinition }

classDefinition ::=
    validClassNames ( "|" validClassNames ) *

validClassNames ::=
    "NON_DEVICE"
    | "DISK_DEVICE"
    | "VIDEO_DEVICE"
    | "NETWORK_DEVICE"
    | "INPUT_DEVICE"
    | "ONBOARD_DEVICE"
    | "OTHER_DEVICE"
    | Number

subclassDefinition ::=
    "SETUP_APPLICATION"
    | "GENERAL_APPLICATION"
    | "FRONT_PAGE"
    | "SINGLE_USE"
    | Number
```

**Note:** There are no BEHAVIORS AND RESTRICTION or an Example for this section.

## 2.5 VFR FormSet List Definition

```
vfrFormSetList ::=
(
    vfrFormDefinition
    vfrFormMapDefinition
    vfrStatementImage
    vfrStatementVarStoreLinear
    vfrStatementVarStoreEfi
    vfrStatementVarStoreNameValue
    vfrStatementDefaultStore
    vfrStatementDisableIfFormSet
    vfrStatementSuppressIfFormSet
) *
```

### BEHAVIORS AND RESTRICTIONS:

In the formset, there can be more than one form of variable stores and default stores. The formset can control whether or not to process an individual form by nesting it inside of an **EFI\_IFR\_DISABLE\_IF** expression.

The variable stores and default stores must be defined before being referenced. Example: None.

## 2.6 VFR Default Stores Definition

```
vfrStatementDefaultStore ::=
    "defaultstore" StringIdentifier ","
    "prompt" "=" getStringId
    { ",", "attribute" "=" Number } ";"
```

### BEHAVIORS AND RESTRICTIONS:

**Note:** **attribute** is used to specify the default ID.

It is optional and the Standard Defaults Identifier, **EFI\_HII\_DEFAULT\_CLASS\_STANDARD**, is used if the attribute is not defined. The Manufacturing Defaults Identifier, **EFI\_HII\_DEFAULT\_CLASS\_MANUFACTURING**, and the Safe Defaults Identifier, **EFI\_HII\_DEFAULT\_CLASS\_SAFE**, can be also used if the header file defining them is included. Example:

```
defaultstore MyStandard, prompt = STRING_TOKEN(STR_STANDARD_DEFAULT);
```

## 2.7 VFR Variable Store Definition

### 2.7.1 VFR Buffer Store Definition

```
vfrStatementVarStoreLinear ::=
  "varstore"
  (
    StringIdentifier ","
    "UINT8"    ","
    "UINT16"   ","
    "UINT32"   ","
    "UINT64"   ","
    "EFI_HII_DATE"  ","
    "EFI_HII_TIME"  ","
  )
  { "varid" "=" Number "," }
  "name" "=" StringIdentifier ","
  "guid" "=" guidDefinition ";
```

#### BEHAVIORS AND RESTRICTIONS:

**Note:** *name* and *guid* are used to specify the variable store jointly.

Example:

```
varstore MyData, name = RefName, guid = FORMSET_GUID;
```

### 2.7.2 VFR EFI Variable Store Definition

```
vfrStatementVarStoreEfi ::=
  "efivarstore" StringIdentifier ","
  "attribute" "=" Number ( "|" Number ) * ","
  "name"      "=" getStringId ",",
  "varsize"   "=" Number ",",
  "guid"      "=" getStringId ";
```

#### BEHAVIORS AND RESTRICTIONS:

**Note:** *name* is used to define the EFI variable name, *guid* is used to define the GUID of EFI variable, and *varsize* is used to define the size of variable.

Example:

```
efivarstore EfiVarStore,
  attribute = EFI_VARIABLE_BOOTSERVICE_ACCESS,
  name      = STRING_TOKEN(STR_TOKEN_NULL),
  varsize   = 1,
  guid      = GUID;
```

### 2.7.3 VFR Variable Name Store Definition

```
vfrStatementVarStoreNameValue ::=
  "namevaluevarstore" StringIdentifier ","
  ( "name" "=" getStringId "," )+
  "guid"      "=" guidDefinition ";
```

## BEHAVIORS AND RESTRICTIONS:

Example:

```
namevaluevarstore NameValueVarStore,  
  name = STRING_TOKEN(STR_NAMEVALUE_TABLE_ITEM1),  
  name = STRING_TOKEN(STR_NAMEVALUE_TABLE_ITEM2),  
  name = STRING_TOKEN(STR_NAMEVALUE_TABLE_ITEM3),  
  guid = GUID;
```

## 2.8 VFR FormSet DisableIf Definition

```
vfrStatementDisableIfFormSet ::=  
  "disableif" vfrStatementExpression ","  
    vfrFormSetList  
  "endif" ","
```

**Note:** There are no BEHAVIORS AND RESTRICTION or an Example for this section.

## 2.9 VFR FormSet SuppressIf Definition

```
vfrStatementSuppressIfFormSet ::=  
  "suppressif" vfrStatementExpression ","  
    vfrFormSetList  
  "endif" ","
```

**Note:** There are no BEHAVIORS AND RESTRICTION or an Example for this section.

## 2.10 VFR General Token Definition

### 2.10.1 GUID Definition

```

Hex8 ::=
  "0x" ["0"- "9" "A"- "F" "a"- "f"] {1,8}

Hex4 ::=
  "0x" ["0"- "9" "A"- "F" "a"- "f"] {1,4}

Hex2 ::=
  "0x" ["0"- "9" "A"- "F" "a"- "f"] {1,2}

guidSubDefinition ::=
  Hex2 ", " Hex2 ", " Hex2 ", " Hex2 ", "
  Hex2 ", " Hex2 ", " Hex2 ", " Hex2

guidDefinition ::=
  "{"
  Hex8 ", " Hex4 ", " Hex4 ", "
  (
    "{" guidSubDefinition "}"
    | guidSubDefinition
  )
  "}"

```

#### BEHAVIORS AND RESTRICTIONS:

In practice, the VFR only supports a GUID with C language-style structure. It is defined as two 32-bit values followed by two 16-bit values followed by eight 1-byte values.

### 2.10.2 String & String Identifier Definition

```

StringIdentifier ::=
  ["A"- "Z" "a"- "z" "_" ] ["A"- "Z" "a"- "z" "_" "0"- "9"] *

getStringId ::=
  "STRING_TOKEN" "(" Number ")"

```

**Note:** There are no BEHAVIORS AND RESTRICTION or an Example for this section.

### 2.10.3 Number Definition

```

Number ::=
  ( "0x" ["0"- "9" "A"- "F" "a"- "f"]+ ) | ["0"- "9"]+

```

**Note:** There are no BEHAVIORS AND RESTRICTION or an Example for this section.

### 2.10.4 VFR Statement Header Definition

```

vfrStatementHeader ::=
  "prompt" "=" getStringId ", "
  "help" "=" getStringId ", "

```

**Note:** There are no BEHAVIORS AND RESTRICTION or an Example for this section.



### 2.10.5 VFR Question Header Definition

```

vfrQuestionHeader ::=
{
  "name"      "=" StringIdentifier "," }
{
  "varid"     "=" vfrStorageVarId  "," }
{
  "questionid" "=" Number  "," }
vfrStatementHeader

vfrQuestionHeaderWithNoStorage ::=
{
  "name"      "=" StringIdentifier "," }
{
  "questionid" "=" Number  "," }
vfrStatementHeader

questionheaderFlagsField ::=
  "READ_ONLY"
  "CALLBACK"
  "RESET_REQUIRED"
  "OPTIONS_ONLY"

vfrStorageVarId ::=
( StringIdentifier "[" Number "]" )
|
(
  StringIdentifier
  (
    "." StringIdentifier { "[" Number "]" }
  ) *
)

```

#### BEHAVIORS AND RESTRICTIONS:

**Note:** *questionid* is used to specify the question ID. If it is not defined, the compiler assigns a unique ID automatically.

**Note:** *name* is used to specify the reference name, which is optional.

Example: None.

### 2.10.6 VFR Constant Value Definition

```

vfrConstantValueField ::=
  Number
  "true"
  "false"
  "one"
  "ones"
  "zero"
  Number ":" Number ":" Number
  Number "/" Number "/" Number
  "STRING_TOKEN" "(" Number ")"

```

**Note:** There are no BEHAVIORS AND RESTRICTION or an Example for this section. .

### 2.10.7 VFR Statement Image & Locked Tag Definition

```

vfrImageTag ::=
    "image" "=" getStringId

vfrLockedTag ::=
    "locked"

vfrStatementStatTag ::=
    vfrImageTag | vfrLockedTag

vfrStatementStatTagList ::=
    vfrStatementStatTag ( "," vfrStatementStatTag ) *

```

**Note:** There are no BEHAVIORS AND RESTRICTION or an Example for this section.

## 2.11 VFR Form Definition

```

vfrFormDefinition ::=
    "form" "formid" "=" Number ","
    "title" "=" getStringId ";"
    (
        vfrStatementImage
        vfrStatementLocked
        vfrStatementRules
        vfrStatementDefault
        vfrStatementStat
        vfrStatementQuestions
        vfrStatementConditional
        vfrStatementLabel
        vfrStatementBanner
    ) *
    "endform" ";"

```

#### BEHAVIORS AND RESTRICTIONS:

**Note:** *formid* must be unique for each form statement in a given formset.

Example:

None.

### 2.11.1 VFR Form Map Definition

```
vfrFormMapDefinition ::=
    "formmap" "formid" "=" Number ","
    (
        "maptitle" "=" getStringId ";"
        "mapguid" "=" guidDefinition ";"
    ) *
    (
        vfrStatementImage
        vfrStatementLocked
        vfrStatementRules
        vfrStatementDefault
        vfrStatementStat
        vfrStatementQuestions
        vfrStatementConditional
        vfrStatementLabel
        vfrStatementBanner
    ) *
    "endform" ";"
```

#### BEHAVIORS AND RESTRICTIONS:

**Note:** *formid* must be unique for each form statement in a given formset.

Example:

None.

### 2.11.2 VFR Image Statement Definition

```
vfrStatementImage ::=
    vfrImageTag ";"
```

**Note:** There are no BEHAVIORS AND RESTRICTION or an Example for this section.

### 2.11.3 VFR Locked Statement Definition

```
vfrStatementLocked ::=
    vfrLockedTag ";"
```

**Note:** There are no BEHAVIORS AND RESTRICTION or an Example for this section.

### 2.11.4 VFR Rule Statement Definition

```
vfrStatementRules ::=
    "rule" StringIdentifier ","
    vfrStatementExpression
    "endrule" ";"
```

#### BEHAVIORS AND RESTRICTIONS:

**StringIdentifier** is the name that can be referenced by a question. It should be unique in the formset.

Example:

```
rule MyRule, 1 + 2
endrule;
```

## 2.11.5 VFR Statement Definition

```
vfrStatementStat ::=
    vfrStatementSubTitle
    | vfrStatementStaticText
    | vfrStatementCrossReference
```

**Note:** There are no BEHAVIORS AND RESTRICTION or an Example for this section.

### 2.11.5.1 VFR SubTitle Definition

```
vfrStatementSubTitle ::=
    "subtitle"
    "text" "=" getStringId
    {
        ", " "flags" "=" vfrSubtitleFlags }
    {
        ", " vfrStatementStatTagList } "; "

vfrSubtitleFlags ::=
    subtitleFlagsField ( "|" subtitleFlagsField ) * "; "

subtitleFlagsField ::=
    Number
    | "HORIZONTAL"
```

#### BEHAVIORS AND RESTRICTIONS:

**Note:** *flags* is optional, and the default value is 0.

Example:

```
subtitle
text = STRING_TOKEN(STR_SUBTITLE_TEXT),
flags = HORIZONTAL;
```

### 2.11.5.2 VFR Text Definition

```
vfrStatementStaticText ::=
    "text"
    "help" "=" getStringId ", "
    "text" "=" getStringId
    {
        ", " "text" "=" getStringId }
    {
        ", " "flags" "=" staticTextFlagsField ( "|" staticTextFlagsField ) *
        ", " "key" "=" Number
    }
    {
        ", " vfrStatementStatTagList } "; "

staticTextFlagsField ::=
    Number
    | questionheaderFlagsField
```

#### BEHAVIORS AND RESTRICTIONS:

**Note:** *flags* is optional. The default value is 0.

## VFR Description in BNF

If **EFI\_IFR\_FLAGS\_CALLBACK** is set in **flags** then it will generate an **EFI\_IFR\_ACTION** op-code. Otherwise, it generates the **EFI\_IFR\_TEXT** op-code.

The value of **key** will be used as a question ID. Example:

Generates **EFI\_IFR\_TEXT**:

```
text
  help = STRING_TOKEN(STR_TEXT_TEXT),
  text = STRING_TOKEN(STR_TEXT_TEXT);
```

Generates **EFI\_IFR\_ACTION**:

```
text
  help = STRING_TOKEN(STR_TEXT_TEXT),
  text = STRING_TOKEN(STR_TEXT_TEXT);
  flags = RESET_REQUIRED,
  key   = 0x0001;
```

### 2.11.5.3 VFR Cross Reference Type Statements Definition

```
vfrStatementCrossReference ::=
    vfrStatementGoto
  | vfrStatementResetButton
```

**Note:** There are no BEHAVIORS AND RESTRICTION or an Example for this section.

### 2.11.5.3.1 VFR Goto Statement Definition

```

vfrStatementGoto ::=
    "goto"
    (
        (
            "devicePath"  "=" getStringId ","
            "formsetguid" "=" guidDefinition ","
            "formid"      "=" Number ","
            "question"    "=" Number ","
        )
        |
        (
            "formsetguid" "=" guidDefinition ","
            "formid"      "=" Number ","
            "question"    "=" Number ","
        )
        |
        (
            "formid"      "=" Number ","
            "question"    "="
            (
                StringIdentifier ","
                | Number ","
            )
        )
        |
        (
            Number ","
        )
    )
    vfrQuestionHeaderWithNoStorage
    {
        "," vfrStatementStatTagList }
    {
        "," "flags" "=" vfrGotoFlags }
    {
        "," "key"      "=" Number
    }

vfrGotoFlags ::=
    gotoFlagsField ( "|" gotoFlagsField ) *

gotoFlagsField ::=
    Number
    | questionheaderFlagsField

```

#### BEHAVIORS AND RESTRICTIONS:

The value of **key** will be used as a question ID. Example:

Generate **EFI\_IFR\_REF** without **key**

```

goto 1,
prompt = STRING_TOKEN(STR_GOTO_PROMPT),
help   = STRING_TOKEN(STR_GOTO_HELP);

```

Generate **EFI\_IFR\_REF** with **key**

```

goto 1,
prompt = STRING_TOKEN(STR_GOTO_PROMPT),
help   = STRING_TOKEN(STR_GOTO_HELP);
key    = 0x1234;

```

Generate **EFI\_IFR\_REF2**

```
goto
    formid      = 1,
    question    = QuesttionRef,
    prompt      = STRING_TOKEN(STR_GOTO_PROMPT),
    help        = STRING_TOKEN(STR_GOTO_HELP);
```

Generate **EFI\_IFR\_REF3**

```
goto
    formsetguid = FORMSET_GUID,
    formid      = 1,
    question    = QuesttionRef,
    prompt      = STRING_TOKEN(STR_GOTO_PROMPT),
    help        = STRING_TOKEN(STR_GOTO_HELP);
```

Generate **EFI\_IFR\_REF4**

```
Goto
    devicepath  = STRING_TOKEN(STR_DEVICE_PATH),
    formsetguid = FORMSET_GUID,
    formid      = 1,
    question    = QuesttionRef,
    prompt      = STRING_TOKEN(STR_GOTO_PROMPT),
    help        = STRING_TOKEN(STR_GOTO_HELP);
```

### 2.11.5.3.2 VFR ResetButton Statement Definition

```
vfrStatementResetButton ::=
    "resetbutton"
    "defaultStore" "=" StringIdentifier ","
    vfrStatementHeader ","
    { vfrStatementStatTagList "," }
    "endresetbutton" ";"
```

#### BEHAVIORS AND RESTRICTIONS:

**Note:** *defaultStore* should point to the default store defined before.

Example:

```
resetbutton
    defaultstore = DefaultStoreRef,
    prompt      = STRING_TOKEN(STR_RESET_BUTTON_PROMPT),
    help        = STRING_TOKEN(STR_RESET_BUTTON_HELP),
endresetbutton;
```

### 2.11.6 VFR Question Type Statements Definition

```
vfrStatementQuestions ::=
    vfrStatementBooleanType
    | vfrStatementDate
    | vfrStatementNumericType
    | vfrStatementStringType
    | vfrStatementOrderedList
    | vfrStatementTime
```

**Note:** There are no BEHAVIORS AND RESTRICTION or an Example for this section.

### 2.11.6.1 VFR Question Tag Definition

```
vfrStatementQuestionTag ::=
    vfrStatementStatTag ","
    | vfrStatementInconsistentIf
    | vfrStatementNoSubmitIf
    | vfrStatementDisableIfQuest
    | vfrStatementRefresh
    | vfrStatementVarstoreDevice
```

**Note:** There are no BEHAVIORS AND RESTRICTION or an Example for this section.

#### 2.11.6.1.1 VFR Question Tag InconsistentIf Definition

```
vfrStatementInconsistentIf ::=
    "inconsistentif"
    "prompt" "=" getStringId ","
    vfrStatementExpression
    "endif"
```

##### BEHAVIORS AND RESTRICTIONS:

Can only be used in questions.

Example:

```
checkbox
    name      = MyCheckBox,
    varid     = MyTestData.mField1,
    questionid = 0xcb,
    prompt    = STRING_TOKEN(STR_CHECK_BOX_PROMPT),
    help      = STRING_TOKEN(STR_CHECK_BOX_HELP),
    flags     = CHECKBOX_DEFAULT | CALLBACK,

    inconsistentif
        prompt    = STRING_TOKEN(STR_INCONSISTENT_IF),
        ideqval MyTestData.mField1 == 2007
    endif
endcheckbox;
```

#### 2.11.6.1.2 VFR Question Tag NoSubmitIf Definition

```
vfrStatementNoSubmitIf ::=
    "nosubmitif"
    "prompt" "=" getStringId ","
    vfrStatementExpression
    "endif"
```

##### BEHAVIORS AND RESTRICTIONS:

Can only be used in questions.

Example:



```
checkboxbox
  name      = MyCheckBox,
  varid     = MyTestData.mField1,
  questionid = 0xcb,
  prompt    = STRING_TOKEN(STR_CHECK_BOX_PROMPT),
  help      = STRING_TOKEN(STR_CHECK_BOX_HELP),
  flags     = CHECKBOX_DEFAULT | CALLBACK,

  nosubmitif prompt    = STRING_TOKEN(STR_NOSUBMIT_IF),
  ideqval MyTestData.mField1 == 2007
endif
endcheckboxbox;
```

#### 2.11.6.1.3 VFR Question Tag DisableIf Definition

```
vfrStatementDisableIfQuest ::=
  "disableif" vfrStatementExpression ";"
  vfrStatementQuestionOptionList
  "endif"
```

##### BEHAVIORS AND RESTRICTIONS:

None.

Example:

```
checkboxbox
  name      = MyCheckBox,
  varid     = MyTestData.mField1,
  questionid = 0xcb,
  prompt    = STRING_TOKEN(STR_CHECK_BOX_PROMPT),
  help      = STRING_TOKEN(STR_CHECK_BOX_HELP),
  flags     = CHECKBOX_DEFAULT | CALLBACK,

  disableif
    ideqvallist MyTestData.mField1 == 1 3 5 7;
    refresh interval = 1
  endif
endcheckboxbox;
```

#### 2.11.6.1.4 VFR Question Tag Refresh Definition

```
vfrStatementRefresh ::=
  "refresh" "interval" "=" Number
```

##### BEHAVIORS AND RESTRICTIONS:

Can only be used in questions.

Example:

Please refer to the example of VFR Question VFR Statement Disalbelf Definition.

#### 2.11.6.1.5 VFR Question Tag VarstoreDevice Definition

```
vfrStatementVarstoreDevice ::=
    "varstoredevice" "=" getStringId ","
```

##### BEHAVIORS AND RESTRICTIONS:

Can only be used in questions. Example:

```
checkbox
    name      = MyCheckBox,
    varid     = MyTestData.mField1,
    questionid = 0xcb,
    prompt    = STRING_TOKEN(STR_CHECK_BOX_PROMPT),
    help      = STRING_TOKEN(STR_CHECK_BOX_HELP),
    flags     = CHECKBOX_DEFAULT | CALLBACK,

    varstoredevice = STRING_TOKEN(STR_VARSTOREDEVICE),
endcheckbox;
```

#### 2.11.6.2 VFR Question Tag List Definition

```
vfrStatementQuestionTagList ::=
    ( vfrStatementQuestionTag )*
```

**Note:** There are no BEHAVIORS AND RESTRICTION or an Example for this section.

#### 2.11.6.3 VFR Question Option Tag Definition

```
vfrStatementQuestionOptionTag ::=
    vfrStatementSuppressIfQuest
    | vfrStatementValue
    | vfrStatementDefault
    | vfrStatementOptions
    | vfrStatementRead
    | vfrStatementWrite
```

**Note:** There are no BEHAVIORS AND RESTRICTION or an Example for this section.

#### 2.11.6.3.1 VFR Question SuppressIf Statement Definition

```
vfrStatementSuppressIfQuest ::=
    "suppressif" vfrStatementExpression ";"
    vfrStatementQuestionOptionList
    "endif"
```

**Note:** There are no BEHAVIORS AND RESTRICTION or an Example for this section.

#### 2.11.6.3.2 VFR Default Statement Definition

```
vfrStatementDefault ::=
  "default"
  (
    (
      vfrStatementValue ","
    | "=" vfrConstantValueField ","
    )
    { "defaultstore" "=" StringIdentifier "," }
  )
```

##### BEHAVIORS AND RESTRICTIONS:

It should be used in question definition.

**Note:** *defaultstore* is optional and it points to the default store defined previously. If *defaultstore* is not defined the **EFI\_HII\_DEFAULT\_CLASS\_STANDARD** is assigned.

Example:

```
default = 1,
default value = 1 + 2,
```

#### 2.11.6.3.3 VFR Value Statement Definition

```
vfrStatementValue ::=
  "value" "=" vfrStatementExpression ";"
```

##### BEHAVIORS AND RESTRICTIONS:

None.

Example:

```
Value = 0;
```

#### 2.11.6.3.4 VFR Option Type Statements Definition

```
vfrStatementOptions ::=
  vfrStatementOneOfOption
```

**Note:** There are no BEHAVIORS AND RESTRICTION or an Example for this section.

### 2.11.6.3.5 VFR OneOf Option Statement Definition

```

vfrStatementOneOfOption ::=
    "option"
    "text" "=" getStringId ","
    "value" "=" vfrConstantValueField ","
    "flags" "=" vfrOneOfOptionFlags
    ( "," vfrImageTag )*
    ","

vfrOneOfOptionFlags ::=
    oneofoptionFlagsField ( "|" oneofoptionFlagsField )*

oneofoptionFlagsField ::=
    Number
    | "OPTION_DEFAULT"
    | "OPTION_DEFAULT_MFG"
    | "INTERACTIVE"
    | "RESET_REQUIRED"
    | "DEFAULT"

```

#### BEHAVIORS AND RESTRICTIONS:

An **option** statement is something special: it is used to embellish or describe questions.

These statements can be used to give the possible value and set the default value for questions. In other words, they are not questions but they influence the questions they embellish. Therefore, the options' **flags** are treated as question flags and can accept all values of question **flags**.

Options with the DEFAULT flags can be used to set the default value for questions. For example:

```

    option text = STRING_TOKEN(STR_ONE_OF_TEXT), value = 0x2, flags =
    DEFAULT | RESET_REQUIRED;

```

### 2.11.6.3.6 VFR Read Statement Definition

```

vfrStatementRead ::=
    "read" vfrStatementExpression ";"

```

#### BEHAVIORS AND RESTRICTIONS:

None.

Example:

None

### 2.11.6.3.7 VFR Write Statement Definition

```

vfrStatementWrite ::=
    "write" vfrStatementExpression ";"

```

#### BEHAVIORS AND RESTRICTIONS:

None.

Example:

None

#### 2.11.6.4 VFR Question Tag List Definition

```
vfrStatementQuestionOptionList ::=  
  (  
    vfrStatementQuestionTag  
    | vfrStatementQuestionOptionTag  
  ) *
```

**Note:** There are no BEHAVIORS AND RESTRICTION or an Example for this section.

#### 2.11.6.5 VFR Boolean Type Statement Definition

```
vfrStatementBooleanType ::=  
  vfrStatementCheckBox  
  | vfrStatementAction
```

**Note:** There are no BEHAVIORS AND RESTRICTION or an Example for this section.

##### 2.11.6.5.1 VFR CheckBox Statement Definition

```
vfrStatementCheckBox ::=  
  "checkbox"  
  vfrQuestionHeader ","  
  { "flags" "=" vfrCheckBoxFlags "," }  
  { "key" "=" Number "," }  
  vfrStatementQuestionOptionList  
  "endcheckbox" ";"  
  
vfrCheckBoxFlags ::=  
  checkboxFlagsField ( "|" checkboxFlagsField ) *  
  
checkboxFlagsField ::=  
  Number  
  | "CHECKBOX_DEFAULT"  
  | "CHECKBOX_DEFAULT_MFG"  
  | questionheaderFlagsField
```

#### BEHAVIORS AND RESTRICTIONS:

The value of **key** is used as question ID.

**Note:** **flags** is optional, and the default value is 0.

Example:

```
checkboxbox
  name      = MyCheckBox,
  varid     = MyTestData.mField1,
  prompt    = STRING_TOKEN(STR_CHECK_BOX_PROMPT),
  help      = STRING_TOKEN(STR_CHECK_BOX_HELP),
  flags     = CHECKBOX_DEFAULT | CALLBACK,
  default   = TRUE,
endcheckboxbox;
```

#### 2.11.6.5.2 VFR Action Statement Definition

```
vfrStatementAction :
  "action"
  vfrQuestionHeaderWithNoStorage ","
  { "flags" "=" vfrActionFlags "," }
  "config" "=" getStringId ","
  vfrStatementQuestionTagList
  "endaction" ","

vfrActionFlags ::=
  actionFlagsField ( "|" actionFlagsField ) *

actionFlagsField ::=
  Number
  | questionheaderFlagsField
```

#### BEHAVIORS AND RESTRICTIONS:

**Note:** *flags* is optional, and the default value is 0.

Example:

```
action
  prompt    = STRING_TOKEN(STR_ACTION_PROMPT),
  help      = STRING_TOKEN(STR_ACTION_HELP),
  flags     = CALLBACK,
  config    = STRING_TOKEN(STR_ACTION_CONFIG),
endaction;
```

#### 2.11.6.6 VFR Numeric Type Statements Definition

```
vfrStatementNumericType ::=
  vfrStatementNumeric
  | vfrStatementOneOf
```

**Note:** *There are no BEHAVIORS AND RESTRICTION or an Example for this section.*

### 2.11.6.6.1 VFR Numeric Statement Definition

```

vfrStatementNumeric ::=
    "numeric"
    vfrQuestionHeader ,
    { "flags" "=" vfrNumericFlags "," }
    { "key"      "=" Number "," }
    vfrSetMinMaxStep
    vfrStatementQuestionOptionList
    "endnumeric" ";"

vfrSetMinMaxStep ::=
    "minimum" "=" Number ","
    "maximum" "=" Number ","
    { "step"   "=" Number "," }

vfrNumericFlags ::=
    numericFlagsField ( "|" numericFlagsField ) *

numericFlagsField ::=
    Number
    | "NUMERIC_SIZE_1"
    | "NUMERIC_SIZE_2"
    | "NUMERIC_SIZE_4"
    | "NUMERIC_SIZE_8"
    | "DISPLAY_INT_DEC"
    | "DISPLAY_UINT_DEC"
    | "DISPLAY_UINT_HEX"
    | questionHeaderFlagsField

```

#### BEHAVIORS AND RESTRICTIONS:

**Note:** *flags* is optional, and the default value partly depends on the size of **varid** defined in **vfrQuestionHeader**.

The default display format is **DISPLAY\_UINT\_DEC**.

Example:

```

numeric
  varid = STTestData.mField2,
  prompt = STRING_TOKEN(STR_NUMERIC_PROMPT),
  help   = STRING_TOKEN(STR_NUMERIC_HELP),
  flags  = DISPLAY_UINT_HEX,
  minimum = 0,
  maximum = 300,
  step    = 0,

  default = 175,
endnumeric;

```

#### 2.11.6.6.2 VFR OneOf Statement Definition

```

vfrStatementOneOf :
  "oneof"
  vfrQuestionHeader ,
  { "flags" "=" vfrOneofFlagsField "," }
  { vfrSetMinMaxStep }
  vfrStatementQuestionOptionList
  "endoneof" ";";

vfrOneofFlagsField ::=
  numericFlagsField ( "|" numericFlagsField ) *

```

#### BEHAVIORS AND RESTRICTIONS:

**Note:** *flags* is optional, and the default value partly depends on the size of **varid** defined in **vfrQuestionHeader** syntax.

The flag is defined in VFR Numeric Statement Definition.

Example:

```

oneof
  varid    = STTestData.mField3[0],
  prompt   = STRING_TOKEN(STR_ONE_OF_PROMPT),
  help     = STRING_TOKEN(STR_ONE_OF_HELP),
  flags    = DISPLAY_UINT_DEC,

  option text = STRING_TOKEN(STR_ONE_OF_TEXT1), value = 0x0, flags = 0;
  option text = STRING_TOKEN(STR_ONE_OF_TEXT2), value = 0x1, flags = 0;
  option text = STRING_TOKEN(STR_ONE_OF_TEXT3), value = 0x2, flags =
DEFAULT;
endoneof;

```

#### 2.11.6.7 VFR String Type Statements Definition

```

vfrStatementStringType ::=
  vfrStatementString
  | vfrStatementPassword

```

**Note:** *There are no BEHAVIORS AND RESTRICTION or an Example for this section.*



### 2.11.6.7.1 VFR String Statement Definition

```

vfrStatementString ::=
    "string"
    vfrQuestionHeader ","
    { "flags" "=" vfrStringFlagsField "," }
    { "key"      "=" Number "," }
    "minsize" "=" Number ","
    "maxsize" "=" Number ","
    vfrStatementQuestionOptionList
    "endstring" ";"

vfrStringFlagsField ::=
    stringFlagsField ( "|" stringFlagsField )*

stringFlagsField ::=
    Number
    | "MULTI LINE"
    | questionheaderFlagsField

```

#### BEHAVIORS AND RESTRICTIONS:

**Note:** *flags* is optional, and the default value 0.

Example:

```

    string
    varid      = STTestData.mField1,
    prompt     = STRING_TOKEN(STR_MY_STRING_PROMPT),
    help       = STRING_TOKEN(STR_MY_STRING_HELP),
    flags      = MULTI_LINE,
    minsize    = 6,
    maxsize    = 0x14,
    endstring;

```

### 2.11.6.7.2 VFR Password Statement Definition

```

vfrStatementPassword ::=
    "password"
    vfrQuestionHeader ,
    { "flags" "=" vfrPasswordFlagsField "," }
    { "key"      "=" Number "," }
    "minsize" "=" Number ","
    "maxsize" "=" Number ","
    vfrStatementQuestionOptionList
    "endpassword" ";"

vfrPasswordFlagsField ::=
    passwordFlagsField ( "|" passwordFlagsField )*

passwordFlagsField ::=
    Number
    | questionheaderFlagsField

```

#### BEHAVIORS AND RESTRICTIONS:

**Note:** *flags* is optional, and the default value 0.

The value of **key** will be used as question ID. Example:

```

password
  varid      = STestData.mPrivate.mField2,
  prompt     = STRING_TOKEN(STR_PASSWORD_PROMPT),
  help       = STRING_TOKEN(STR_PASSWORD_HELP),
  minsize    = 6,
  maxsize    = 20,
endpassword;

```

### 2.11.6.8 VFR OrderedList Statement Definition

```

vfrStatementOrderedList ::=
  "orderedlist"
  vfrQuestionHeader ","
  { "maxcontainers" "=" Number "," }
  { "flags" "=" vfrOrderedListFlags }
  vfrStatementQuestionOptionList
  "endlist" ";"

vfrOrderedListFlags ::=
  orderedlistFlagsField ( "|" orderedlistFlagsField )*

orderedlistFlagsField ::=
  Number
  | "UNIQUE"
  | "NOEMPTY"
  | questionheaderFlagsField

```

#### BEHAVIORS AND RESTRICTIONS:

**Note:** *maxcontainers* is optional, and the default value depends on the variable size defined by *varid* in *vfrQuestionHeader*.

**Note:** *flags* is optional, and the default value 0.

Example:

```

orderedlist
  varid      = STesPriData.mField3,
  prompt     = STRING_TOKEN(STR_BOOT_OPTIONS),
  help       = STRING_TOKEN(STR_BOOT_OPTIONS),

  option text = STRING_TOKEN(STR_BOOT_OPTION2), value = 2, flags =
RESET_REQUIRED;
  option text = STRING_TOKEN(STR_BOOT_OPTION1), value = 1, flags =
RESET_REQUIRED;
  option text = STRING_TOKEN(STR_BOOT_OPTION3), value = 3, flags =
RESET_REQUIRED;
  option text = STRING_TOKEN(STR_BOOT_OPTION4), value = 4, flags =
RESET_REQUIRED;
endlist;

```

### 2.11.6.9 VFR Date Statement Definition

```

vfrStatementDate ::=
  "date"
  (
    (
      vfrQuestionHeader ,
      { "flags" "=" vfrDateFlags "," }
      vfrStatementQuestionOptionList
    )
    |
    (
      "year" "varid" "=" StringIdentifier "." StringIdentifier ","
      "prompt" "=" getStringId ","
      "help"    "=" getStringId ","
      minMaxDateStepDefault

      "month" "varid" "=" StringIdentifier "." StringIdentifier ","
      "prompt" "=" getStringId ","
      "help"    "=" getStringId ","
      minMaxDateStepDefault

      "day" "varid" "=" StringIdentifier "." StringIdentifier ","
      "prompt" "=" getStringId ","
      "help"    "=" getStringId ","
      minMaxDateStepDefault

      ( vfrStatementInconsistentIf ) *
    )
  )
  "enddate" ","

minMaxDateStepDefault ::=
  "minimum"    "=" Number ","
  "maximum"    "=" Number ","
  { "step"      "=" Number "," }
  { "default"   "=" Number "," }

vfrDateFlags ::=
  dateFlagsField ( "|" dateFlagsField ) *

dateFlagsField ::=
  Number
  |
  "YEAR_SUPPRESS"
  |
  "MONTH_SUPPRESS"
  |
  "DAY_SUPPRESS"
  |
  "STORAGE_NORMAL"
  |
  "STORAGE_TIME"
  |
  "STORAGE_WAKEUP"

```

**BEHAVIORS AND RESTRICTIONS:**

There are old and new syntax of VFR Date statement they are incompatible; the old VFR Date only use EFI Date/Time Storage instead of normal question value storage; the new VFR Date can use either the normal question value storage or EFI Date/Time Storage.

For new syntax of VFR, **flags** is optional, and the default value is 0. Examples follow.

New syntax of VFR Date:

```
date
  varid    = STestData.mDate,
  prompt   = STRING_TOKEN(STR_DATE_PROMPT),
  help     = STRING_TOKEN(STR_DATE_PROMPT),
  flags    = STORAGE_NORMAL,
  default  = 2007/08/26,
enddate;
```

Old syntax of VFR Date:

```
date    year varid = Date.Year,
        prompt    = STRING_TOKEN(STR_DATE_PROMPT),
        help      = STRING_TOKEN(STR_DATE_YEAR_HELP),
        minimum   = 2003,
        maximum   = 2100,
        step      = 1,
        default   = 2003,

        month varid = Date.Month
        prompt      = STRING_TOKEN(STR_DATE_PROMPT),
        help        = STRING_TOKEN(STR_DATE_MONTH_HELP),
        minimum     = 1,
        maximum     = 12,
        step        = 1,
        default     = 1,

        day varid  = Date.Day,
        prompt     = STRING_TOKEN(STR_DATE_PROMPT),
        help       = STRING_TOKEN(STR_DATE_DAY_HELP),
        minimum    = 1,
        maximum    = 31,
        step       = 0x1,
        default    = 1,
enddate;
```

### 2.11.6.10 VFR Time Statement Definition

```

vfrStatementTime :
    "time"
    (
        (
            vfrQuestionHeader ","
            { "flags" "=" vfrTimeFlags "," }
            vfrStatementDefault
        )
        |
        (
            "hour" "varid" "=" StringIdentifier "." StringIdentifier ","
            "prompt" "=" getStringId ","
            "help"   "=" getStringId ","
            minMaxTimeStepDefault

            "minute" "varid" "=" StringIdentifier "." StringIdentifier ","
            "prompt"  "=" getStringId ","
            "help"    "=" getStringId ","
            minMaxTimeStepDefault

            "second" "varid" "=" StringIdentifier "." StringIdentifier ","
            "prompt"  "=" getStringId ","
            "help"    "=" getStringId ","
            minMaxTimeStepDefault
        )
    )
    "endtime" ","

minMaxTimeStepDefault ::=
    "minimum"  "=" Number ","
    "maximum"  "=" Number ","
    { "step"    "=" Number "," }
    { "default" "=" Number "," }

vfrTimeFlags ::=
    timeFlagsField ( "|" timeFlagsField ) *

timeFlagsField ::=
    Number
    | "HOUR_SUPPRESS"
    | "MINUTE_SUPPRESS"
    | "SECOND_SUPPRESS"
    | "STORAGE_NORMAL"
    | "STORAGE_TIME"
    | "STORAGE_WAKEUP"

```

#### BEHAVIORS AND RESTRICTIONS:

There are old and new syntax of VFR Time statement they are incompatible; the old VFR Date only use EFI Date/Time Storage instead of normal question value storage; the new VFR Time can use either the normal question value storage or EFI Date/Time Storage.

For new syntax of VFR, **flags** is optional, and the default value is 0. Example:

New Time Syntax:

```

time
    name      = MyTime,
    varid     = STestData.mTime,
    prompt    = STRING_TOKEN(STR_TIME_PROMPT),
    help      = STRING_TOKEN(STR_TIME_PROMPT),
    flags     = STORAGE_NORMAL,
    default   = 15:33:33,
endtime;

```

Old Time Syntax:

```

time hour varid = Time.Hours,
    prompt      = STRING_TOKEN(STR_TIME_PROMPT),
    help        = STRING_TOKEN(STR_TIME_HOUR_HELP),
    minimum     = 0,
    maximum     = 23,
    step        = 1,
    default     = 0,

    minute varid = Time.Minutes,
    prompt      = STRING_TOKEN(STR_TIME_PROMPT),
    help        = STRING_TOKEN(STR_TIME_MINUTE_HELP),
    minimum     = 0,
    maximum     = 59,
    step        = 1,
    default     = 0,

    second varid = Time.Seconds,
    prompt      = STRING_TOKEN(STR_TIME_PROMPT),
    help        = STRING_TOKEN(STR_TIME_SECOND_HELP),
    minimum     = 0,
    maximum     = 59,
    step        = 1,
    default     = 0,
endtime;

```

## 2.11.7 VFR Conditional Type Statements Definition

```

vfrStatementConditional ::=
    vfrStatementDisableIfStat
    | vfrStatementSuppressIfStat
    | vfrStatementGrayOutIfStat

```

**Note:** There are no BEHAVIORS AND RESTRICTION or an Example for this section.

### 2.11.7.1 VFR Statement List Definition

```

vfrStatementStatList ::=
    vfrStatementStat
    | vfrStatementQuestions
    | vfrStatementConditional
    | vfrStatementLabel

```

**Note:** There are no BEHAVIORS AND RESTRICTION or an Example for this section.

### 2.11.7.2 VFR Statement Disalbel f Definition

```

vfrStatementDisableIfStat ::=
    "disableif" vfrStatementExpression ";"
    ( vfrStatementStatList ) *
    "endif" ";"

```

**Note:** There are no BEHAVIORS AND RESTRICTION or an Example for this section.

### 2.11.7.3 VFR Statement SuppressIf Definition

```
vfrStatementSuppressIfStat ::=
    "suppressif" vfrStatementExpression ";"
    ( vfrStatementStatList ) *
    "endif" ";"
```

**Note:** There are no BEHAVIORS AND RESTRICTION or an Example for this section.

### 2.11.7.4 VFR Statement GrayOutIf Definition

```
vfrStatementGrayOutIfStat ::=
    "grayoutif" vfrStatementExpression ";"
    ( vfrStatementStatList ) *
    "endif" ";"
```

**Note:** There are no BEHAVIORS AND RESTRICTION or an Example for this section.

## 2.11.8 VFR GUID Statement Definition

### 2.11.8.1 VFR Label Statement Definition

```
vfrStatementLabel :
    "label" Number ";"
```

#### BEHAVIORS AND RESTRICTIONS:

It is an extended **EFI\_IFR\_GUID** op-code.

It should be paired in the VFR program.

**Note:** **label** is used to insert op-codes at runtime.

Example:

```
label LABEL_START;
label LABEL_END;
```

### 2.11.8.2 VFR Banner Statement Definition

```
vfrStatementBanner :
    "banner" { "," }
    "title" "=" getStringId ","
    (
        (
            "line" Number ","
            "align" ( "left" | "center" | "right" ) ";"
        )
        |
        ( "timeout" "=" Number ";" )
    )
```

#### BEHAVIORS AND RESTRICTIONS:

It is an extended **EFI\_IFR\_GUID** op-code.

Example:

```

banner
    title = STRING_TOKEN(STR_BANNER_TITLE),
    line 1,
    align center;

```

## 2.12 VFR Expression Statement Definition

### BEHAVIORS AND RESTRICTIONS:

The VFR expression is defined as C language style.

The syntax of VFR expression is defined as a tree. The positions in the tree are determined according to the priority of the operator (for example: `+` `-` `*` `/`). At the root it is the terms of **OR** and followed by the terms of **AND** because the priority of operator **OR** is lower than the operator **AND**(s). The leaves of the tree are sub-expressions of built-in-functions, unary operators, ternary operators and constant.

### 2.12.1 OR

```

vfrStatementExpression ::=
    andTerm ( "OR" andTerm ) *

```

### BEHAVIORS AND RESTRICTIONS:

Generate **EFI\_IFR\_OR** op-codes.

### 2.12.2 AND

```

andTerm ::=
    bitwiseorTerm ( "AND" bitwiseorTerm ) *

```

### BEHAVIORS AND RESTRICTIONS:

Generate **EFI\_IFR\_AND** op-codes.

### 2.12.3 bitwiseor

```

bitwiseorTerm ::=
    bitwiseandTerm ( "|" bitwiseandTerm ) *

```

### BEHAVIORS AND RESTRICTIONS:

Generate **EFI\_IFR\_BITWISE\_OR** op-codes.

### 2.12.4 bitwiseand

```

bitwiseandTerm ::=
    equalTerm ( "&" equalTerm ) *

```

### BEHAVIORS AND RESTRICTIONS:

Generate **EFI\_IFR\_BITWISE\_AND** op-codes.



### 2.12.5 equal

```
equalTerm ::=
  compareTerm
  (
    "==" compareTerm
    | "!=" compareTerm
  ) *
```

#### BEHAVIORS AND RESTRICTIONS:

Generate **EFI\_IFR\_EQUAL** or **EFI\_IFR\_NOT\_EQUAL** op-codes.

### 2.12.6 compare

```
compareTerm ::=
  shiftTerm
  (
    "<" shiftTerm
    | "<=" shiftTerm
    | ">" shiftTerm
    | ">=" shiftTerm
  ) *
```

#### BEHAVIORS AND RESTRICTIONS:

Generate **EFI\_IFR\_LESS\_THAN** **EFI\_IFR\_LESS\_EQUAL** **EFI\_IFR\_GREATER\_EQUAL** or **EFI\_IFR\_GREATER\_THAN** op-codes.

### 2.12.7 shift

```
shiftTerm ::=
  addMinusTerm
  (
    "<<" addMinusTerm
    | ">>" addMinusTerm
  ) *
```

#### BEHAVIORS AND RESTRICTIONS:

Generate **EFI\_IFR\_SHIFT\_LEFT** or **EFI\_IFR\_SHIFT\_RIGHT** op-codes.

### 2.12.8 add/minus

```
addMinusTerm ::=
  multdivmodTerm
  (
    "+" multdivmodTerm
    | "-" multdivmodTerm
  ) *
```

#### BEHAVIORS AND RESTRICTIONS:

Generate **EFI\_IFR\_ADD** or **EFI\_IFR\_SUBTRACT** op-codes.

## 2.12.9 multiply/divide/modulo

```
multdivmodTerm ::=
  castTerm
  (
    "*" castTerm
    | "/" castTerm
    | "%" castTerm
  ) *
```

### BEHAVIORS AND RESTRICTIONS:

Generate **EFI\_IFR\_MULTIPLY** **EFI\_IFR\_MODULO** or **EFI\_IFR\_DIVIDE** op-codes.

## 2.12.10 cast terms

```
castTerm ::=
  (
    " ("
    (
      "BOOLEAN"
      | "UINT64"
      | "UINT32"
      | "UINT16"
      | "UINT8"
    )
  ) " "
  ) *
  atomTerm
```

### BEHAVIORS AND RESTRICTIONS:

The VFR supports the C-style type conversion. The values can be converted into one of four types: **BOOLEAN** **UINT64** **UINT32** **UINT16** **UINT8**.

## 2.12.11 atom terms

```
atomTerm ::=
  vfrExpressionCatenate
  | vfrExpressionMatch
  | vfrExpressionParen
  | vfrExpressionBuildInFunction
  | vfrExpressionConstant
  | vfrExpressionUnaryOp
  | vfrExpressionTernaryOp
  | vfrExpressionMap
  ( "NOT" atomTerm )
```

### 2.12.11.1 catenate

```
vfrExpressionCatenate ::=
  "catenate"
  "(" vfrStatementExpression "," vfrStatementExpression ")"
```

### BEHAVIORS AND RESTRICTIONS:

Generate **EFI\_IFR\_CATENATE** op-codes.

#### 2.12.11.2 match

```
vfrExpressionMatch ::=  
    "match"  
    "(" vfrStatementExpression "," vfrStatementExpression ")"
```

##### BEHAVIORS AND RESTRICTIONS:

Generate **EFI\_IFR\_MATCH** op-codes.

#### 2.12.11.3 parenthesis

```
vfrExpressionParen ::=  
    "(" vfrStatementExpression ")"
```

##### BEHAVIORS AND RESTRICTIONS:

Change the order of the calculation.

#### 2.12.11.4 build-in functions

```
vfrExpressionBuildInFunction ::=  
    dupExp  
    | ideqvalExp  
    | ideqidExp  
    | ideqvallistExp  
    | questionref13Exp  
    | rulerefExp  
    | stringref1Exp  
    | pushtthisExp  
    | securityExp  
    | getExp
```

##### 2.12.11.4.1 dup

```
dupExp ::=  
    "dup"
```

##### BEHAVIORS AND RESTRICTIONS:

Generate **EFI\_IFR\_DUP** op-codes.

##### 2.12.11.4.2 ideqval

```
ideqvalExp ::=  
    "ideqval"  
    vfrQuestionDataFieldName "==" Number
```

##### BEHAVIORS AND RESTRICTIONS:

Generate **EFI\_IFR\_EQ\_ID\_VAL** op-codes.

#### 2.12.11.4.3 ideqid

```
ideqidExp ::=
  "ideqid"
  vfrQuestionDataFieldName "==" vfrQuestionDataFieldName
```

##### BEHAVIORS AND RESTRICTIONS:

Generate **EFI\_IFR\_EQ\_ID\_ID** op-codes.

#### 2.12.11.4.4 ideqvallist

```
ideqvallistExp ::=
  "ideqvallist"
  vfrQuestionDataFieldName "==" ( Number )+
```

##### BEHAVIORS AND RESTRICTIONS:

Generate **EFI\_IFR\_EQ\_ID\_LIST** op-codes.

#### 2.12.11.4.5 questionref

```
questionref13Exp ::=
  "questionref"
  (
    (
      { "path" "=" getStringId }
      { "guid" "=" guidDefinition }
    )
    |
    (
      "(" StringIdentifier | Number ")"
    )
  )
```

##### BEHAVIORS AND RESTRICTIONS:

Generate **EFI\_IFR\_QUESTION\_REF1** **EFI\_IFR\_QUESTION\_REF3\_2** or **EFI\_IFR\_QUESTION\_REF3\_3** op-codes. Examples follow.

Generate **EFI\_IFR\_QUESTION\_REF1**:  
**questionref** (MyCheckBox)

Generate **EFI\_IFR\_QUESTION\_REF3\_2**:  
**questionref** path = STRING\_TOKEN (STR\_DEVICE\_PATH) (MyCheckBox)

Generate **EFI\_IFR\_QUESTION\_REF3\_3**:

```
questionref
  path = STRING_TOKEN (STR_DEVICE_PATH)
  guid = GUID (MyCheckBox)
```

#### 2.12.11.4.6 ruleref

```
rulerefExp ::=
  "ruleref" "(" StringIdentifier ")"
```

##### BEHAVIORS AND RESTRICTIONS:

Generate **EFI\_IFR\_RULE\_REF** op-codes.

#### 2.12.11.4.7 stringref

```
stringreflExp ::=
  "stringref" "(" getStringId ")"
```

##### BEHAVIORS AND RESTRICTIONS:

Generate **EFI\_IFR\_RULE\_REF** op-codes.

#### 2.12.11.4.8 pushthis

```
pushthisExp ::=
  "pushthis"
```

##### BEHAVIORS AND RESTRICTIONS:

Generate **EFI\_IFR\_THIS** op-codes.

#### 2.12.11.4.9 security

```
securityExp ::=
  "security" "(" guidDefinition ")"
```

##### BEHAVIORS AND RESTRICTIONS:

Generate **EFI\_IFR\_SECURITY** op-codes.

#### 2.12.11.4.10get

```
getExp ::=
  "get" "(" vfrStorageVarId { "|" "flags" "=" vfrNumericFlags } ")"
```

##### BEHAVIORS AND RESTRICTIONS:

Generate **EFI\_IFR\_GET** op-codes.

### 2.12.11.5 constant

```
vfrExpressionConstant ::=
    "true"
    "false"
    "one"
    "ones"
    "zero"
    "undefined"
    "version"
    Number
```

#### BEHAVIORS AND RESTRICTIONS:

Generate **EFI\_IFR\_TRUE** **EFI\_IFR\_FALSE** **EFI\_IFR\_ONE** **EFI\_IFR\_ONES** **EFI\_IFR\_ZERO** **EFI\_IFR\_UNDEFINED** or **EFI\_IFR\_VERSION** op-codes.

### 2.12.11.6 unary operators

```
vfrExpressionUnaryOp ::=
    lengthExp
    bitwisenotExp
    question2refExp
    stringref2Exp
    toboolExp
    uintExp
    toupperExp
    tolowerExp
    setExp
```

#### 2.12.11.6.1 length

```
lengthExp ::=
    "length" "(" vfrStatementExpression ")"
```

#### BEHAVIORS AND RESTRICTIONS:

Generate **EFI\_IFR\_LENGTH** op-codes.

#### 2.12.11.6.2 bitwisenot

```
bitwisenotExp ::=
    "bitwisenot" "(" vfrStatementExpression ")"
```

#### BEHAVIORS AND RESTRICTIONS:

Generate **EFI\_IFR\_BITWISENOT** op-codes.

#### 2.12.11.6.3 questionrefval

```
question2refExp ::=
    "questionrefval" "(" vfrStatementExpression ")"
```

#### BEHAVIORS AND RESTRICTIONS:

Generate **EFI\_IFR\_QUESTION\_REF2** op-codes.

#### 2.12.11.6.4 stringrefval

```
stringref2Exp ::=  
    "stringrefval" "(" vfrStatementExpression ")"
```

##### BEHAVIORS AND RESTRICTIONS:

Generate **EFI\_IFR\_STRING\_REF2** op-codes.

#### 2.12.11.6.5 boolval

```
toboolExp ::=  
    "boolval" "(" vfrStatementExpression ")"
```

##### BEHAVIORS AND RESTRICTIONS:

Generate **EFI\_IFR\_TO\_BOOLEAN** op-codes.

#### 2.12.11.6.6 stringval

```
tostringExp ::=  
    "stringval" { "format" "=" Number "," }  
    "(" vfrStatementExpression ")"
```

##### BEHAVIORS AND RESTRICTIONS:

Generate **EFI\_IFR\_TO\_STRING** op-codes.

#### 2.12.11.6.7 uintval

```
uintExp ::=  
    "uintval" "(" vfrStatementExpression ")"
```

##### BEHAVIORS AND RESTRICTIONS:

Generate **EFI\_IFR\_TO\_UINT** op-codes.

#### 2.12.11.6.8 toupper

```
toupperExp ::=  
    "toupper" "(" vfrStatementExpression ")"
```

##### BEHAVIORS AND RESTRICTIONS:

Generate **EFI\_IFR\_TO\_UPPER** op-codes.

#### 2.12.11.6.9 tolower

```
tolowerExp ::=  
    "tolower" "(" vfrStatementExpression ")"
```

##### BEHAVIORS AND RESTRICTIONS:

Generate **EFI\_IFR\_TO\_LOWER** op-codes.

### 2.12.11.6.10 set

```
setExp ::=
  "set"
  "("
  vfrStorageVarId { "|" "flags" "=" vfrNumericFlags } ","
  vfrStatementExpression
  ")"
```

#### BEHAVIORS AND RESTRICTIONS:

Generate **EFI\_IFR\_SET** op-codes.

### 2.12.11.7 ternary operators

```
vfrExpressionTernaryOp ::=
  conditionalExp
  | findExp
  | midExp
  | tokenExp
  | spanExp
```

#### 2.12.11.7.1 cond

```
conditionalExp ::=
  "cond"
  "("
  vfrStatementExpression
  "?"
  vfrStatementExpression
  ":"
  vfrStatementExpression
  ")"
```

#### BEHAVIORS AND RESTRICTIONS:

Generate **EFI\_IFR\_CONDITIONAL** op-codes.

#### 2.12.11.7.2 find

```
findExp ::=
  "find"
  "("
  findFormat ( "|" findFormat ) *
  ","
  vfrStatementExpression
  ","
  vfrStatementExpression
  ","
  vfrStatementExpression
  ")"

findFormat ::=
  "SENSITIVE"
  | "INSENSITIVE"
```

#### BEHAVIORS AND RESTRICTIONS:

Generate **EFI\_IFR\_FIND** op-codes.



#### 2.12.11.7.3 mid

```
midExp ::=
    "mid"
    "("
    vfrStatementExpression
    ","
    vfrStatementExpression
    ","
    vfrStatementExpression
    ")"
```

##### BEHAVIORS AND RESTRICTIONS:

Generate **EFI\_IFR\_MID** op-codes.

#### 2.12.11.7.4 tok

```
tokenExp ::=
    "tok"
    "("
    vfrStatementExpression
    ","
    vfrStatementExpression
    ","
    vfrStatementExpression
    ")"
```

##### BEHAVIORS AND RESTRICTIONS:

Generate **EFI\_IFR\_TOKEN** op-codes.

#### 2.12.11.7.5 span

```
spanExp ::=
    "span"
    "("
    "flags" "=" spanFlags ( "|" spanFlags ) *
    ","
    vfrStatementExpression
    ","
    vfrStatementExpression
    ","
    vfrStatementExpression
    ")"

spanFlags ::=
    Number
    | "LAST_NON_MATCH"
    | "FIRST_NON_MATCH"
```

##### BEHAVIORS AND RESTRICTIONS:

Generate **EFI\_IFR\_SPAN** op-codes.

### 2.12.11.8 map

```

vfrExpressionMap ::=
    "map"
    "("
    vfrStatementExpression
    ":"
    (
        vfrStatementExpression
        ","
        vfrStatementExpression
        ";"
    ) *
    ")"

```

#### BEHAVIORS AND RESTRICTIONS:

Generate **EFI\_IFR\_MAP** op-codes.