Johar Interface Interpreters (IntIs)
Requirements Specification for All Java IntIs

This document describes the requirements that must be met by all Johar interface interpreters (IntIs). It is assuming that the IntI is written in Java, but similar requirements apply to IntIs written in any language.

In this document, the words *must* and *must not* are in italics and indicate normative statements (strict requirements for the IntI). The words *may* and *does not have to* are also in italics, but indicate behaviour that is permitted or not forbidden; they are used for clarity. The words *should* and *should not* are also in italics, and indicate behaviour that is recommended but not necessary.

The requirements in this document are numbered 1, 2, 3 and so on. They will be referred to in other documents as IntI-R1, IntI-R2, IntI-R3, and so on.

1 Core Steps

This section describes the sequence of steps that must be taken by an IntI in interacting with the IDF, the `GemSetting`, and the application engine. We refer to these as the "core steps".

1. A given IntI *may* take many steps other than the core steps, and *may* interact with the user in many ways outside of these core steps (for instance, in providing help or facilitating table browsing).
2. However, whenever it interacts with the IDF, the `GemSetting` and the application engine, the method calls *must* follow the pattern of the core steps.

Core steps:

3. Initialization phase:
   (a) The IntI *must* first get a `johar.idf.Idf` object by using the method `johar.idf.Idf.idfFromFile(fname)`, where the file name argument is the name of an XML or IDF-format file.
   (b) If `johar.idf.Idf.idfFromFile(fname)` throws an `IdfFormatException`, then the IntI *must* show the error message and exit.
   (c) If the IDF's `IdfVersion` is greater than that supported by the IntI, then the IntI *must* show an error message and exit.
   (d) The IntI *must* then call `GemFactory.newGemSetting()`, using the `Idf` object returned by `idfFromFile` as the first parameter and a valid *ShowTextHandler* as the second parameter.
   (e) The IntI *must* then call `GemSetting.validate()` on the `GemSetting` returned by `GemFactory`.
   (f) If `GemSetting.validate()` throws an `IdfFormatException`, the IntI *must* show the error message and exit.
   (g) The IntI *must* then `GemSetting.validate()` on the `GemSetting` returned by `GemFactory`.
   (h) The IntI *must* then call `GemSetting.initializeAppEngine()`. (This will set up the `Gem` tables and call the `InitializationMethod` for the app engine.)
4. After the initialization phase, the IntI *may* continue with the processing described below.

5. The IntI *may* then perform the Command Loop (see below) as many times as needed, until the IntI terminates.

6. At any time while executing the Command Loop, the IntI *may* exit its current iteration and begin a new iteration. [Rationale: the user may cancel the command processing at any time.]

The Command Loop consists of the following steps:

7. The IntI *may* call the `ActiveIfMethod`s of any commands that have them.

8. It *must* then call `GemSetting.selectCurrentCommand(cmdName)`. If `cmdName` corresponds to a command with an `ActiveIfMethod`, then that method *must* be one that was called since the beginning of this iteration of the Command Loop. [Rationale: the active status of a command may have been changed by the effect of the previous command.]

9. It *must* then call `GemSetting.selectCurrentStage(0)`. [Rationale: the parameter values for one stage must be loaded and checked before the default values of the parameters in the next stage are obtained, to ensure correct communication with the application engine methods of multi-stage commands.]

10. The IntI *may* then perform the Stage Loop (see below) as many times as needed, at least until `MinNumberOfReps` repetitions of of every parameter of every stage of the command has been loaded into the `GemSetting`.

11. The IntI *may* continue to perform the Stage Loop after a value for every parameter of every stage of the command has been loaded.

12. The IntI *must* continue to perform the Stage Loop until every `ParameterCheckMethod`, in every stage of the command that has one, has returned `null` or the empty string.

13. The IntI *must* perform the following steps for every question in the current command, from the first question specified to the last question specified.

    (a) The IntI *must* call the `AskIfMethod` of the question.

    (b) If the `AskIfMethod` returns `true`, and the question has a `DefaultValueMethod`, the IntI *must* call the `DefaultValueMethod` of the question.

    (c) If the `AskIfMethod` returns `true`, the IntI *must* load a value for the question into the `GemSetting`.

14. The IntI *must* call the `CommandMethod` for the command.

15. The IntI *must* ensure that the `showText` text from the command has been effectively communicated to the user. [Rationale: the `showText` from the last command executed in a run of the application may be important, and must not be rendered inaccessible by the termination of the application.]

16. If the command's `QuitAfter` attribute is `false`, but the command has a `QuitAfterIfMethod`, then the IntI *must* call that method.

17. If the command's `QuitAfter` attribute is `true`, or the command has a `QuitAfterIfMethod` which has returned `true`, then the IntI *must* terminate.

See Table 1 for the Java type that a Java IntI *must* load as the value of a parameter, depending on what `Type` the parameter is.

Parameter `Type`	Java type	Comment
`boolean`	`boolean`	
`choice`	`java.lang.String`	One of the choices
`date`	`java.util.Calendar`	
`file`	`java.io.File`	
`float`	`double`	The floating-point type with the maximum range and precision in Java
`int`	`long`	The integer type with the maximum range and precision in Java
`text`	`java.lang.String`	
`tableEntry`	`int`	The row number (starting with 0) of one row that the user has selected
`timeOfDay`	`java.util.Calendar`	

Figure 1: Bindings of Johar parameter types to Java types.

The Stage Loop consists of the following steps:

18. At any time, the IntI *may* call the `DefaultValueMethod` of any parameter in the current stage.
19. At any time, the IntI *may* load values for repetitions of any parameter in the current stage.
20. If the IntI loads a value for a repetition of a parameter with a `DefaultValue` or `DefaultValueMethod`, and the value was not selected directly by the user, then the value *must* come from the `DefaultValue` or from a call to the `DefaultValueMethod` made since the beginning of this iteration of the Stage Loop. [Rationale: the previous stage's `ParameterCheckMethod` may have caused a change to the value returned by the parameter's `DefaultValueMethod`.]
21. The IntI *must* load at least `MinNumberOfReps` repetitions of each parameter in the current stage before calling the current stage's `ParameterCheckMethod`, unless the parameter has a `ParentParameter` whose value is not the parameter's `ParentValue`.
22. At any time after that, the IntI *may* call the current stage's `ParameterCheckMethod`, if it has one.
23. If the current stage has a `ParameterCheckMethod`, then the IntI *must* call it at least once.
24. If the current stage has a `ParameterCheckMethod`, then the IntI *must not* load any parameter values between the time it last calls the `ParameterCheckMethod` and the time it next calls `GemSetting.selectCurrentStage()`.
25. Finally, the IntI *may* call `GemSetting.selectCurrentStage()` with a parameter that is either one greater than the index number of the current stage, or less than the index number of the current stage.

2 Other Requirements

Requirements concerning values of parameters and questions:

26. The IntI *must* provide a way for the user to select values for parameters and questions for any command.

27. For a parameter or question of type `choice`, any value loaded by the IntI *must* be a choice from the parameter's or question's `Choices` string.

28. For a parameter or question of type `file`, any value loaded by the IntI *must* respect the parameter's or question's `FileConstraint`.

29. For a parameter or question of type `text`, any value loaded by the IntI *must* respect the parameter's or question's `MaxNumberOfChars` and `MaxNumberOfLines` attribute values.

30. For a parameter or question of type `int` or `float`, any value loaded by the IntI *must* respect the parameter's or question's `MaxValue` and `MinValue` attribute values.

Requirements concerning repetitions of parameters:

31. The IntI *must* provide a way for the user to select multiple values for repetitions of every parameter of a command, up to the parameter's `MaxNumberOfReps`.

32. For every parameter, the IntI *must* load at least `MinNumberOfReps` repetitions for the parameter before calling the `CommandMethod` of the command, unless the parameter has a `ParentParameter` whose value is not the `ParentValue` of the parameter.

33. For every parameter, the IntI *must not* load a value for the parameter if the parameter has a `ParentParameter` whose value is not the `ParentValue` of the parameter.

34. For every parameter, the IntI *must* load at most `MaxNumberOfReps` repetitions for the parameter before calling the `CommandMethod` of the command.

35. If the `RepsModel` of a parameter is `set`, then the IntI *may* load only one repetition of each value selected by the user.

36. If the `RepsModel` of a parameter is `multiset` or `sequence`, then the IntI *must* load the number of repetitions of each value selected by the user.

37. If the `RepsModel` of a parameter is `set` or `multiset`, then the IntI *may* load values in any order.

38. If the `RepsModel` of a parameter is `sequence`, then the IntI *must* load values in the order specified by the user.

Requirements concerning user interface structure:

39. The IntI *may* use the value of `Application` as a unique identifier of the application currently being run.

40. The IntI *may* use the application's command groups in order to structure the interface.

41. The IntI *may* use the `Prominence` of commands, etc. in order to structure the interface.

42. The IntI *should* use the `Label` of a command, parameter, or question as a description, where needed in the interface.

43. The IntI *may* use the `RepsModel` of any parameter in order to structure the interface.

44. The IntI *should* provide a convenient way for the user to select valid values for parameters or questions of type `date`, `file`, and `timeOfDay`.

Other IntI requirements:

45. The IntI *must* provide access to all commands in the application.
46. For a command with any stage with any parameter of type `tableEntry`, where the `SourceTable` is browsable, the IntI *must not* select the command as the current command unless the user has selected at least `MinNumberOfReps` entries in the parameter's `SourceTable`.
47. The IntI *must* allow the user to access, browse and select rows in all browsable tables that are not currently hidden.
48. The IntI *must not* allow the user to access, browse or select rows in non-browsable tables.
49. The IntI *must not* allow the user to access, browse or select rows in tables that are currently hidden.
50. The IntI *must* provide access to all the help messages provided in the IDF.
51. The IntI *may* use any of the `Label`, `BriefHelp`, `OneLineHelp`, and `MultiLineHelp` messages provided in the IDF wherever they are needed.