Format of .met and .nav Files (inferred)

Eskarina of Morningthaw/Coldeve

September 20, 2021

Contents

List of Tables

1 Introduction

While creating the `metaf` tool, I methodically and exhaustively worked through .met and .nav files to reverse-engineer the formats of their contents. The information in this document summarizes my key conclusions. I make no promises as to its accuracy, though it does seem quite workable. Note that while I include some `metaf` terminology in a couple of places, this is just for informational convenience in linking up to it, as its own documentation may help augment the information described here. This is a .met and .nav format document, though—not a `metaf` document.

In general, .met files are formatted as sets of tables, either in sequence or nested inside each other. They require sequential reading to consume their contents, so if there's a formatting error at some point, usually all subsequent data cannot be properly read due to its misalignment with formatting expectations.

2 A Quick Word on the Broader Format

.met files represent a special case of a more generalized file-format system also used in VirindiTank's character profiles and gameinfo database. Years ago, Virindi provided a short description of its structure. Here it is, in summary.

Files are:

```
TableCount #number of tables saved in file
[Tables] #multiple lines; iterated to define TableCount number of tables
```

Tables are:

```
TableName #name of table
ColumnCount #number of columns in table
[ColumnNames] #ColumnCount lines, defining ColumnCount column names
[IsColumnIndexed] #ColumnCount lines, specifying if each column is indexed (y or n)
RecordCount #number of records (rows) in the table
[Records] #multiple lines; iterated to define RecordCount number of records
```

Virindi stated that column indexing changes nothing in the file; it just meant that that column cannot have duplicates, and it told VirindiTank to build a runtime index for that column.

Records (rows) are comprised of a ColumnCount-long series of pairs of fields (the combination of which I call a pair-field), which are:

```
DataTypeCode #specifies type of data on the following line(s)
DataValue #the data itself, which may be multiple lines or a single line
```

The possible DataTypeCodes are listed in Table 1.

Table 1: All DataTypeCodes in the general format.

DataTypeCode	**Meaning**
d	double
i	integer
u	unsigned integer
f	float
s	string
b	boolean
0	unassigned (null)
ba	length-prefixed ASCII blob (called ByteArray below)
TABLE	nested database table, encoded the same as above (minus TableName)

The .met file format uses this general formatting system in a restricted way, and the remainder of this document focuses on those specifics.

3 Meta

The Meta section of a .met file is the first data section encountered. Its key parameter affecting the entire remainder of the file is the RuleCount on file line 14, which states how many Rules are defined in the file (zero or more). See Section 4 for Rule format.

```
 1 1 #TableCount (only one meta in the file)
 2 CondAct #TableName (ConditionAction (and State))
 3 5 #ColumnCount
 4 CType #Column1 Name (the ConditionType)
 5 AType #Column2 Name (the ActionType)
 6 CData #Column3 Name (the ConditionData)
 7 AData #Column4 Name (the ActionData)
 8 State #Column5 Name (the StateName)
 9 n #Column1 (CType) not indexed
10 n #Column2 (AType) not indexed
11 n #Column3 (CData) not indexed
12 n #Column4 (AData) not indexed
13 n #Column5 (State) not indexed
14 integer #RuleCount (RecordCount)
15 [Rules] #multiple lines; iterated to define RuleCount number of Rules (Records)
```

4 Rule

A Rule represents a single line in the main VirindiTank meta interface. It contains exactly one Condition (which may be simple or compound) and exactly one Action (which also may be simple or compound). See Section 5 and Section 6 for more on Condition and Action formats, respectively.

Note that ConditionType and ActionType pair-fields (the pair of 'i' and ID lines for each) are separated from their corresponding ConditionData and ActionData pair-fields by intervening text when they're specified at the top level of a Rule definition. However, when nested inside a compound Condition, the ConditionType and ConditionData blocks end up contiguous; likewise for nested Actions. This is because a Rule places the ConditionType, ActionType, ConditionData, and ActionData into columns CType, AType, CData, and AData, respectively, whereas a compound Condition requires the use of an All or Any or Not, each one of which places the ConditionType and ConditionData into their internal columns, K and V (see Section 5.4, Section 5.5, and Section 5.23); likewise for nested Actions, with ActionType and ActionData being placed into the All (Action) internal columns, K and V (see Section 6.5).

```
i #integer DataTypeCode for ConditionType pair-field (CType column)
integer #ConditionID DataValue for ConditionType pair-field (CType column)
i #integer DataTypeCode for ActionType pair-field (AType column)
integer #ActionID DataValue for ActionType pair-field (AType column)
ConditionData #multiple lines; simple or compound (via All/Any/Not) (CData column)
ActionData #multiple lines; simple or compound (via All (Action)) (AData column)
s #string DataTypeCode for StateName pair-field (State column)
string #StateName for this Rule DataValue for StateName pair-field (State column)
```

5 Condition

Conditions are defined in the left-side pane when creating a new rule in a meta, in VirindiTank. See Table 2 for a summary of all ConditionID codes.

5.1 Unassigned (Condition) (unavailable)

ConditionID -1. Rumored Condition, but I never saw it, nor is it available in VirindiTank. I don't recall where I heard this code means this. But regardless, I don't know the details of its format.

5.2 Never

ConditionID 0. Never True. Heed the red note at the start of Section 4.

ConditionType and ConditionData:

```
i #integer DataTypeCode for ConditionType (or K) pair-field
0 #ConditionID DataValue for ConditionType (or K) pair-field
```

```
i #integer DataTypeCode for ConditionData (or V) pair-field
0 #DataValue (null) for ConditionData (or V) pair-field
```

5.3 Always

ConditionID 1. Always True. Heed the red note at the start of Section 4.

ConditionType and ConditionData:

```
i #integer DataTypeCode for ConditionType (or K) pair-field
1 #ConditionID DataValue for ConditionType (or K) pair-field
```

```
i #integer DataTypeCode for ConditionData (or V) pair-field
0 #DataValue (null) for ConditionData (or V) pair-field
```

5.4 All (Condition)

ConditionID 2. True if all Conditions inside it are True. (True if empty.) Do not confuse with the 'All' Action (Section 6.5). 'All' may contain zero or more Conditions inside it, as indicated in ConditionCount (here shown on line 7 of the ConditionData); it performs a logical-AND operation to combine all its inner Conditions into a single Condition. The last lines (8 and 9) are iterated to define ConditionCount number of Conditions (each of which is multiple lines itself). It is important to note that this means arbitrarily complex Conditions may be built because these inner Conditions may themselves be All, Any, or Not, which in turn may contain yet more Conditions (and so on). Heed the red note at the start of Section 4.

ConditionType and ConditionData:

```
i #integer DataTypeCode for ConditionType (or K) pair-field
2 #ConditionID DataValue for ConditionType (or K) pair-field
```

```
1 TABLE #TABLE DataTypeCode for ConditionData (or V) pair-field
2 2 #ColumnCount for this TABLE
3 K #Column1 Name
4 V #Column2 Name
5 n #Column1 (K) is not indexed
6 n #Column2 (V) is not indexed
7 integer #ConditionCount, for iterating the following two pair-fields to define zero or
      more  nested Conditions (RecordCount)
8 [ConditionType] #pair-field (K column)
9 [ConditionData] #pair-field; simple or compound (via All/Any/Not) (V column)
```

5.5 Any

ConditionID 3. True if any Condition inside it is True. (True if empty.) 'Any' may contain zero or more Conditions inside it, as indicated in ConditionCount (here shown on line 7 of the ConditionData); it performs a logical-OR operation to combine all its inner Conditions into a single Condition. The last lines (8 and 9) are iterated to define ConditionCount number of Conditions (each of which is multiple lines itself). It is important to note that this means arbitrarily complex Conditions may be built because these inner Conditions may themselves be All, Any, or Not, which in turn may contain yet more Conditions (and so on). Heed the red note at the start of Section 4.

ConditionType and ConditionData:

```
i #integer DataTypeCode for ConditionType (or K) pair-field
3 #ConditionID DataValue for ConditionType (or K) pair-field
```

```
TABLE #TABLE DataTypeCode for ConditionData (or V) pair-field
2 #ColumnCount for this TABLE
K #Column1 Name
V #Column2 Name
n #Column1 (K) is not indexed
n #Column2 (V) is not indexed
integer #ConditionCount, for iterating the following two pair-fields to define zero or
    more  nested Conditions (RecordCount)
[ConditionType] #pair-field (K column)
[ConditionData] #pair-field; simple or compound (via All/Any/Not) (V column)
```

5.6 Chat Message

ConditionID 4. True if ChatPattern matches a chat-window message. (Matches any message if left empty.) Heed the red note at the start of Section 4.

ConditionType and ConditionData:

```
i #integer DataTypeCode for ConditionType (or K) pair-field
4 #ConditionID DataValue for ConditionType (or K) pair-field
```

```
s #string DataTypeCode for ConditionData (or V) pair-field
string #ChatPattern (regular expression) #DataValue for ConditionData (or V) pair-field
```

5.7 Pack Slots <=

ConditionID 5. True if number of empty slots remaining in character's main pack inventory is <= SlotCount. Heed the red note at the start of Section 4.

ConditionType and ConditionData:

```
i #integer DataTypeCode for ConditionType (or K) pair-field
5 #ConditionID DataValue for ConditionType (or K) pair-field
```

```
i #integer DataTypeCode for ConditionData (or V) pair-field
integer #SlotCount #DataValue for ConditionData (or V) pair-field
```

5.8 Seconds in state >=

ConditionID 6. True if time elapsed since entering current state is >= Seconds. Resets timer if meta is stopped/started. Heed the red note at the start of Section 4.

ConditionType and ConditionData:

```
i #integer DataTypeCode for ConditionType (or K) pair-field
6 #ConditionID DataValue for ConditionType (or K) pair-field
```

```
i #integer DataTypeCode for ConditionData (or V) pair-field
integer #Seconds #DataValue for ConditionData (or V) pair-field
```

5.9 Navroute empty

ConditionID 7. True if current nav route is empty. Heed the red note at the start of Section 4.

ConditionType and ConditionData:

```
i #integer DataTypeCode for ConditionType (or K) pair-field
7 #ConditionID DataValue for ConditionType (or K) pair-field
```

```
i #integer DataTypeCode for ConditionData (or V) pair-field
0 #DataValue (null) for ConditionData (or V) pair-field
```

5.10 Character Death

ConditionID 8. True if character death detected. Heed the red note at the start of Section 4.

ConditionType and ConditionData:

```
i #integer DataTypeCode for ConditionType (or K) pair-field
8 #ConditionID DataValue for ConditionType (or K) pair-field
```

```
i #integer DataTypeCode for ConditionData (or V) pair-field
0 #DataValue (null) for ConditionData (or V) pair-field
```

5.11 Any Vendor Open

ConditionID 9. True when any vendor window is opened. Heed the red note at the start of Section 4.

ConditionType and ConditionData:

```
i #integer DataTypeCode for ConditionType (or K) pair-field
9 #ConditionID DataValue for ConditionType (or K) pair-field
```

```
i #integer DataTypeCode for ConditionData (or V) pair-field
0 #DataValue (null) for ConditionData (or V) pair-field
```

5.12 Vendor Closed

ConditionID 10. True when a vendor window is closed. Heed the red note at the start of Section 4.

ConditionType and ConditionData:

```
i #integer DataTypeCode for ConditionType (or K) pair-field
10 #ConditionID DataValue for ConditionType (or K) pair-field
```

```
i #integer DataTypeCode for ConditionData (or V) pair-field
0 #DataValue (null) for ConditionData (or V) pair-field
```

5.13 Inventory Item Count <=

ConditionID 11. True if number of ItemName in inventory is <= ItemCount. Heed the red note at the start of Section 4.

ConditionType and ConditionData:

```
i #integer DataTypeCode for ConditionType (or K) pair-field
11 #ConditionID DataValue for ConditionType (or K) pair-field
```

```
TABLE #TABLE DataTypeCode for ConditionData (or V) pair-field
2 #ColumnCount for this TABLE
k #Column1 Name
v #Column2 Name
n #Column1 (k) is not indexed
n #Column2 (v) is not indexed
2 #RecordCount
s #string DataTypeCode for pair-field (k column)
n #'item [n]ame'? DataValue for pair-field (k column)
s #string DataTypeCode for pair-field (v column)
string #ItemName (DataValue for pair-field (v column))
s #string DataTypeCode for pair-field (k column)
c #'item [c]ount'? DataValue for pair-field (k column)
i #integer DataTypeCode for pair-field (v column)
integer #ItemCount (DataValue for pair-field (v column))
```

5.14 Inventory Item Count >=

ConditionID 12. True if number of ItemName in inventory is >= ItemCount. Heed the red note at the start of Section 4.

ConditionType and ConditionData:

```
i #integer DataTypeCode for ConditionType (or K) pair-field
12 #ConditionID DataValue for ConditionType (or K) pair-field
```

```
TABLE #TABLE DataTypeCode for ConditionData (or V) pair-field
2 #ColumnCount for this TABLE
k #Column1 Name
v #Column2 Name
n #Column1 (k) is not indexed
n #Column2 (v) is not indexed
2 #RecordCount
s #string DataTypeCode for pair-field (k column)
n #‘item [n]ame’? DataValue for pair-field (k column)
s #string DataTypeCode for pair-field (v column)
string #ItemName (DataValue for pair-field (v column))
s #string DataTypeCode for pair-field (k column)
c #‘item [c]ount’? DataValue for pair-field (k column)
i #integer DataTypeCode for pair-field (v column)
integer #ItemCount (DataValue for pair-field (v column))
```

5.15 Monster Name Count Within Distance

ConditionID 13. True if number of MonsterName matches within Distance is >= MonsterCount. Heed the red note at the start of Section 4.

ConditionType and ConditionData:

```
i #integer DataTypeCode for ConditionType (or K) pair-field
13 #ConditionID DataValue for ConditionType (or K) pair-field
```

```
TABLE #TABLE DataTypeCode for ConditionData (or V) pair-field
2 #ColumnCount for this TABLE
k #Column1 Name
v #Column2 Name
n #Column1 (k) is not indexed
n #Column2 (v) is not indexed
3 #RecordCount
s #string DataTypeCode for pair-field (k column)
n #‘regex [n]ame of monster’? DataValue for pair-field (k column)
s #string DataTypeCode for pair-field (v column)
string #MonsterName (regular expression) (DataValue for pair-field (v column))
s #string DataTypeCode for pair-field (k column)
c #‘monster [c]ount’? DataValue for pair-field (k column)
i #integer DataTypeCode for pair-field (v column)
integer #MonsterCount (DataValue for pair-field (v column))
s #string DataTypeCode for pair-field (k column)
r #‘[r]ange’? DataValue for pair-field (k column)
d #double DataTypeCode for pair-field (v column)
double #Distance (DataValue for pair-field (v column))
```

5.16 Monster Priority Count Within Distance

ConditionID 14. True if number of exactly-MonsterPriority monsters within Distance is >= MonsterCount. Heed the red note at the start of Section 4.

ConditionType and ConditionData:

```
i #integer DataTypeCode for ConditionType (or K) pair-field
14 #ConditionID DataValue for ConditionType (or K) pair-field
```

```
TABLE #TABLE DataTypeCode for ConditionData (or V) pair-field
2 #ColumnCount for this TABLE
k #Column1 Name
v #Column2 Name
n #Column1 (k) is not indexed
n #Column2 (v) is not indexed
3 #RecordCount
s #string DataTypeCode for pair-field (k column)
p #'monster [p]riority'? DataValue for pair-field (k column)
i #integer DataTypeCode for pair-field (v column)
integer #MonsterPriority (DataValue for pair-field (v column))
s #string DataTypeCode for pair-field (k column)
c #'monster [c]ount'? DataValue for pair-field (k column)
i #integer DataTypeCode for pair-field (v column)
integer #MonsterCount (DataValue for pair-field (v column))
s #string DataTypeCode for pair-field (k column)
r #'[r]ange'? DataValue for pair-field (k column)
d #double DataTypeCode for pair-field (v column)
double #Distance (DataValue for pair-field (v column))
```

5.17 Need to Buff

ConditionID 15. True if VirindiTank's settings determine character needs to buff. Heed the red note at the start of Section 4.

ConditionType and ConditionData:

```
i #integer DataTypeCode for ConditionType (or K) pair-field
15 #ConditionID DataValue for ConditionType (or K) pair-field
```

```
i #integer DataTypeCode for ConditionData (or V) pair-field
0 #DataValue (null) for ConditionData (or V) pair-field
```

5.18 No Monsters Within Distance

ConditionID 16. True if there are no monsters within Distance of character. Heed the red note at the start of Section 4.

ConditionType and ConditionData:

```
i #integer DataTypeCode for ConditionType (or K) pair-field
16 #ConditionID DataValue for ConditionType (or K) pair-field
```

```
TABLE #TABLE DataTypeCode for ConditionData (or V) pair-field
2 #ColumnCount for this TABLE
k #Column1 Name
v #Column2 Name
n #Column1 (k) is not indexed
n #Column2 (v) is not indexed
1 #RecordCount
s #string DataTypeCode for pair-field (k column)
r #‘[r]ange’? DataValue for pair-field (k column)
d #double DataTypeCode for pair-field (v column)
double #Distance (DataValue for pair-field (v column))
```

5.19 Landblock ==

ConditionID 17. True if character location is currently in Landblock. Heed the red note at the start of Section 4.

ConditionType and ConditionData:

```
i #integer DataTypeCode for ConditionType (or K) pair-field
17 #ConditionID DataValue for ConditionType (or K) pair-field
```

```
i #integer DataTypeCode for ConditionData (or V) pair-field
integer #Landblock #DataValue for ConditionData (or V) pair-field
```

5.20 Landcell ==

ConditionID 18. True if character location is currently in Landcell. Heed the red note at the start of Section 4.

ConditionType and ConditionData:

```
i #integer DataTypeCode for ConditionType (or K) pair-field
18 #ConditionID DataValue for ConditionType (or K) pair-field
```

```
i #integer DataTypeCode for ConditionData (or V) pair-field
integer #Landcell #DataValue for ConditionData (or V) pair-field
```

5.21 Portalspace Entered

ConditionID 19. True upon entering portalspace. Heed the red note at the start of Section 4.

ConditionType and ConditionData:

```
i #integer DataTypeCode for ConditionType (or K) pair-field
19 #ConditionID DataValue for ConditionType (or K) pair-field
```

```
i #integer DataTypeCode for ConditionData (or V) pair-field
0 #DataValue (null) for ConditionData (or V) pair-field
```

5.22 Portalspace Exited

ConditionID 20. True upon exiting portalspace. Heed the red note at the start of Section 4.

ConditionType and ConditionData:

```
i #integer DataTypeCode for ConditionType (or K) pair-field
20 #ConditionID DataValue for ConditionType (or K) pair-field
```

```
i #integer DataTypeCode for ConditionData (or V) pair-field
0 #DataValue (null) for ConditionData (or V) pair-field
```

5.23 Not

ConditionID 21. True if Condition inside it is False. ‘Not’ must contain exactly one Condition inside it (as indicated on line 7 of the ConditionData, where All and Any define ConditionCount) but otherwise shares the same format structure as All and Any (Section 5.4 and Section 5.5). It performs a logical-NOT on the Condition inside it (which may contain yet more Conditions if it’s an All, Any, or Not). Heed the red note at the start of Section 4.

ConditionType and ConditionData:

```
i #integer DataTypeCode for ConditionType (or K) pair-field
21 #ConditionID DataValue for ConditionType (or K) pair-field
```

```
1 TABLE #TABLE DataTypeCode for ConditionData (or V) pair-field
2 2 #ColumnCount for this TABLE
3 K #Column1 Name
4 V #Column2 Name
5 n #Column1 (K) is not indexed
6 n #Column2 (V) is not indexed
7 1 #ConditionCount for Not must be 1 (RecordCount)
8 ConditionType #pair-field (K column)
9 ConditionData #pair-field; simple or compound (via All/Any/Not) (V column)
```

5.24 Seconds in state (P) >=

ConditionID 22. True if time elapsed since entering current state is >= Seconds. Persistent timer; does not reset if meta is stopped/started. Heed the red note at the start of Section 4.

ConditionType and ConditionData:

```
i #integer DataTypeCode for ConditionType (or K) pair-field
22 #ConditionID DataValue for ConditionType (or K) pair-field
```

```
i #integer DataTypeCode for ConditionData (or V) pair-field
integer #Seconds #DataValue for ConditionData (or V) pair-field
```

5.25 Time Left On Spell >=

ConditionID 23. True if time remaining on spell with SpellID is >= Seconds. Heed the red note at the start of Section 4.

ConditionType and ConditionData:

```
i #integer DataTypeCode for ConditionType (or K) pair-field
23 #ConditionID DataValue for ConditionType (or K) pair-field
```

```
TABLE #TABLE DataTypeCode for ConditionData (or V) pair-field
2 #ColumnCount for this TABLE
k #Column1 Name
v #Column2 Name
n #Column1 (k) is not indexed
n #Column2 (v) is not indexed
2 #RecordCount
s #string DataTypeCode for pair-field (k column)
sid #‘[s]pell[id]’? DataValue for pair-field (k column)
i #integer DataTypeCode for pair-field (v column)
integer #SpellID (DataValue for pair-field (v column))
s #string DataTypeCode for pair-field (k column)
sec #‘[sec]onds’? DataValue for pair-field (k column)
i #integer DataTypeCode for pair-field (v column)
integer #Seconds (DataValue for pair-field (v column))
```

5.26 Burden Percent >=

ConditionID 24. True if character burden percent is >= BurdenPercent. Heed the red note at the start of Section 4.

ConditionType and ConditionData:

```
i #integer DataTypeCode for ConditionType (or K) pair-field
24 #ConditionID DataValue for ConditionType (or K) pair-field
```

```
i #integer DataTypeCode for ConditionData (or V) pair-field
integer #BurdenPercent #DataValue for ConditionData (or V) pair-field
```

5.27 Dist any route pt >=

ConditionID 25. True if character's shortest-distance to current nav route is >= Distance (in yards). Heed the red note at the start of Section 4.

ConditionType and ConditionData:

```
i #integer DataTypeCode for ConditionType (or K) pair-field
25 #ConditionID DataValue for ConditionType (or K) pair-field
```

```
TABLE #TABLE DataTypeCode for ConditionData (or V) pair-field
2 #ColumnCount for this TABLE
k #Column1 Name
v #Column2 Name
n #Column1 (k) is not indexed
n #Column2 (v) is not indexed
1 #RecordCount
s #string DataTypeCode for pair-field (k column)
dist #'[dist]ance'? DataValue for pair-field (k column)
d #double DataTypeCode for pair-field (v column)
double #Distance (DataValue for pair-field (v column))
```

5.28 Expression

ConditionID 26. True if Expression evaluates to True. (Do not confuse this with the Actions 'Expression Action' (Section 6.9) and 'Chat Expression' (Section 6.10).) Heed the red note at the start of Section 4.

ConditionType and ConditionData:

```
i #integer DataTypeCode for ConditionType (or K) pair-field
26 #ConditionID DataValue for ConditionType (or K) pair-field
```

```
TABLE #TABLE DataTypeCode for ConditionData (or V) pair-field
2 #ColumnCount for this TABLE
k #Column1 Name
v #Column2 Name
n #Column1 (k) is not indexed
n #Column2 (v) is not indexed
1 #RecordCount
s #string DataTypeCode for pair-field (k column)
e #'[e]xpression'? DataValue for pair-field (k column)
s #string DataTypeCode for pair-field (v column)
string #Expression (DataValue for pair-field (v column))
```

5.29 ClientDialogPopup (unavailable)

ConditionID 27. Rumored Condition, but I never saw it, nor is it available in VirindiTank. It may not even be defined, or perhaps it was but isn't anymore? I don't even remember where I heard of this ostensibly being defined. I recommend never trying to use it. Details of format are unknown to me.

5.30 Chat Message Capture

ConditionID 28. True if both ChatPattern matches a chat-window message and that message's 'color channel' is in the ColorIdList (or that list is empty). Enables chat snippets to be saved into variables using regular expressions. See `metaf` documentation of 'ChatCapture' for more on the ColorIdList. Heed the red note at the start of Section 4.

ConditionType and ConditionData:

```
i #integer DataTypeCode for ConditionType (or K) pair-field
28 #ConditionID DataValue for ConditionType (or K) pair-field
```

```
TABLE #TABLE DataTypeCode for ConditionData (or V) pair-field
2 #ColumnCount for this TABLE
k #Column1 Name
v #Column2 Name
n #Column1 (k) is not indexed
n #Column2 (v) is not indexed
2 #RecordCount
s #string DataTypeCode for pair-field (k column)
p #'regex [p]attern'? DataValue for pair-field (k column)
s #string DataTypeCode for pair-field (v column)
string #ChatPattern (regular expression) (DataValue for pair-field (v column))
s #string DataTypeCode for pair-field (k column)
c #'[c]olor id list'? DataValue for pair-field (k column)
s #string DataTypeCode for pair-field (v column)
string #ColorIdList (semicolon-delimited 'color channel' number list) (DataValue (v column))
```

6 Action

Actions are defined in the right-side pane when creating a new rule in a meta, in VirindiTank. See Table 3 for a summary of all ActionID codes.

6.1 Unassigned (Action)

ActionID -1. Rumored Action, but I never saw it, nor is it available in VirindiTank. I don't recall where I heard this code means this. But regardless, the details of the format are unknown to me.

6.2 None

ActionID 0. Do nothing. (Action: none.) Heed the red note at the start of Section 4.

ActionType and ActionData:

```
i #integer DataTypeCode for ActionType (or K) pair-field
0 #ActionID DataValue for ActionType (or K) pair-field
```

```
i #integer DataTypeCode for ActionData (or V) pair-field
0 #DataValue (null) for ActionData (or V) pair-field
```

6.3 Set Meta State

ActionID 1. Set current state to StateName. Heed the red note at the start of Section 4.

ActionType and ActionData:

```
i #integer DataTypeCode for ActionType (or K) pair-field
1 #ActionID DataValue for ActionType (or K) pair-field
```

```
s #string DataTypeCode for ActionData (or V) pair-field
string #StateName #DataValue for ActionData (or V) pair-field
```

6.4 Chat Command

ActionID 2. Send ChatCommand as chat. Do not confuse this with the 'Chat Expression' Action (Section 6.10). Heed the red note at the start of Section 4.

ActionType and ActionData:

```
i #integer DataTypeCode for ActionType (or K) pair-field
2 #ActionID DataValue for ActionType (or K) pair-field
```

```
s #string DataTypeCode for ActionData (or V) pair-field
string #ChatCommand #DataValue for ActionData (or V) pair-field
```

6.5 All (Action)

ActionID 3. Do not confuse this with the 'All' Condition (Section 5.4). 'All' (Action) may contain zero or more Actions inside it, as indicated in ActionCount (here shown on line 7); it simply serves as a container to hold multiple Actions in response to a satisfied Condition. The last lines (8 and 9) are iterated to define ActionCount number of Actions (each of which is multiple lines itself). The Action may be another All (Action), though there's not really any reason to 'nest' Actions like that. Heed the red note at the start of Section 4.

ActionType and ActionData:

```
i #integer DataTypeCode for ActionType (or K) pair-field
3 #ActionID DataValue for ActionType (or K) pair-field
```

```
1 TABLE #TABLE DataTypeCode for ActionData (or V) pair-field
2 2 #ColumnCount for this TABLE
3 K #Column1 Name
4 V #Column2 Name
5 n #Column1 (K) is not indexed
6 n #Column2 (V) is not indexed
7 integer #ActionCount, for iterating the following two pair-fields to define zero or more
      nested Actions (RecordCount)
8 [ActionType] #pair-field (K column)
9 [ActionData] #pair-field; simple or compound (though there's not much point) (V column)
```

6.6 Load Embedded Nav Route

ActionID 4. Enables embedding nav routes inside metas; the ByteArray (which is exactly ByteCount characters (bytes) in length, including any newline characters encountered) contains all the nav route information inside it, including its in-meta name and nav route node count. See Section 7 for more on interpreting the ByteArray as an embedded nav route. Heed the red note at the start of Section 4.

ActionType and ActionData:

```
i #integer DataTypeCode for ActionType (or K) pair-field
4 #ActionID DataValue for ActionType (or K) pair-field
```

```
ba #ByteArray DataTypeCode for ActionData (or V) pair-field
integer #ByteCount (length prefix for ByteArray)
ByteArray #run of ByteCount chars, incl. any newlines #DataValue for ActionData (or V) pair-field
```

6.7 Call Meta State

ActionID 5. Sets state to CallStateName, placing ReturnToStateName on the ‘call stack’ in order to remember which state to go to when ‘Return From Call’ (Section 6.8) is encountered. Keep calls and returns in careful balance. Heed the red note at the start of Section 4.

ActionType and ActionData:

```
i #integer DataTypeCode for ActionType (or K) pair-field
5 #ActionID DataValue for ActionType (or K) pair-field
```

```
TABLE #TABLE DataTypeCode for ActionData (or V) pair-field
2 #ColumnCount for this TABLE
k #Column1 Name
v #Column2 Name
n #Column1 (k) is not indexed
n #Column2 (v) is not indexed
2 #RecordCount
s #string DataTypeCode for pair-field (k column)
st #‘set [st]ate’? DataValue for pair-field (k column)
s #string DataTypeCode for pair-field (v column)
string #CallStateName (DataValue for pair-field (v column))
s #string DataTypeCode for pair-field (k column)
ret #‘[ret]urn to state’? DataValue for pair-field (k column)
s #string DataTypeCode for pair-field (v column)
string #ReturnToStateName (DataValue for pair-field (v column))
```

6.8 Return From Call

ActionID 6. Expects a state to be on the ‘call stack’ because this Action pops it and sets the current state to it. (See Section 6.7.) Keep calls and returns in careful balance. Heed the red note at the start of Section 4.

ActionType and ActionData:

```
i #integer DataTypeCode for ActionType (or K) pair-field
6 #ActionID DataValue for ActionType (or K) pair-field
```

```
i #integer DataTypeCode for ActionData (or V) pair-field
0 #DataValue (null) for ActionData (or V) pair-field
```

6.9 Expression Action

ActionID 7. Evaluates ExpressionAction. Do not confuse this with the Condition ‘Expression’ (Section 5.28), or the Action ‘Chat Expression’ (Section 6.10). Heed the red note at the start of Section 4.

ActionType and ActionData:

```
i #integer DataTypeCode for ActionType (or K) pair-field
7 #ActionID DataValue for ActionType (or K) pair-field
```

```
TABLE #TABLE DataTypeCode for ActionData (or V) pair-field
2 #ColumnCount for this TABLE
k #Column1 Name
v #Column2 Name
n #Column1 (k) is not indexed
n #Column2 (v) is not indexed
1 #RecordCount
s #string DataTypeCode for pair-field (k column)
e #‘[e]xpression action’? DataValue for pair-field (k column)
s #string DataTypeCode for pair-field (v column)
string #ExpressionAction (DataValue for pair-field (v column))
```

6.10 Chat Expression

ActionID 8. Evaluates ChatExpression, then sends the result as chat. Do not confuse this with the Condition ‘Expression’ (Section 5.28), or the Actions ‘Chat Command’ (Section 6.4) or ‘Expression Action’ (Section 6.9). Heed the red note at the start of Section 4.

ActionType and ActionData:

```
i #integer DataTypeCode for ActionType (or K) pair-field
8 #ActionID DataValue for ActionType (or K) pair-field
```

```
TABLE #TABLE DataTypeCode for ActionData (or V) pair-field
2 #ColumnCount for this TABLE
k #Column1 Name
v #Column2 Name
n #Column1 (k) is not indexed
n #Column2 (v) is not indexed
1 #RecordCount
s #string DataTypeCode for pair-field (k column)
e #‘chat [e]xpression’? DataValue for pair-field (k column)
s #string DataTypeCode for pair-field (v column)
string #ChatExpression (DataValue for pair-field (v column))
```

6.11 Set Watchdog

ActionID 9. Creates a watchdog in the current state that is activated if at any time while still in that state your character has not moved >= Distance over the preceding Seconds of time. If triggered, StateName is called. (Returning from it, re-enters the original state.) Leaving the current state deletes this watchdog, as does the ‘Clear Watchdog’ Action (Section 6.12). Heed the red note at the start of Section 4.

ActionType and ActionData:

```
i #integer DataTypeCode for ActionType (or K) pair-field
9 #ActionID DataValue for ActionType (or K) pair-field
```

```
TABLE #TABLE DataTypeCode for ActionData (or V) pair-field
2 #ColumnCount for this TABLE
k #Column1 Name
v #Column2 Name
n #Column1 (k) is not indexed
n #Column2 (v) is not indexed
3 #RecordCount
s #string DataTypeCode for pair-field (k column)
s #‘[s]tate name’? DataValue for pair-field (k column)
s #string DataTypeCode for pair-field (v column)
string #StateName (DataValue for pair-field (v column))
s #string DataTypeCode for pair-field (k column)
r #‘[r]ange’? DataValue for pair-field (k column)
d #double DataTypeCode for pair-field (v column)
double #Distance (DataValue for pair-field (v column))
s #string DataTypeCode for pair-field (k column)
t #‘[t]ime’? DataValue for pair-field (k column)
d #double DataTypeCode for pair-field (v column)
double #Seconds (DataValue for pair-field (v column))
```

6.12 Clear Watchdog

ActionID 10. Deletes the current state's watchdog, as created by the 'Set Watchdog' Action (Section 6.11). Heed the red note at the start of Section 4.

ActionType and ActionData:

```
i #integer DataTypeCode for ActionType (or K) pair-field
10 #ActionID DataValue for ActionType (or K) pair-field
```

```
TABLE #TABLE DataTypeCode for ActionData (or V) pair-field
2 #ColumnCount for this TABLE
k #Column1 Name
v #Column2 Name
n #Column1 (k) is not indexed
n #Column2 (v) is not indexed
0 #RecordCount
```

6.13 Get VT Option

ActionID 11. Gets the current value of the VirindiTank option OptionName and saves it in variable VariableName. (Also see Section 6.14.) Heed the red note at the start of Section 4.

ActionType and ActionData:

```
i #integer DataTypeCode for ActionType (or K) pair-field
11 #ActionID DataValue for ActionType (or K) pair-field
```

```
TABLE #TABLE DataTypeCode for ActionData (or V) pair-field
2 #ColumnCount for this TABLE
k #Column1 Name
v #Column2 Name
n #Column1 (k) is not indexed
n #Column2 (v) is not indexed
2 #RecordCount
s #string DataTypeCode for pair-field (k column)
o #'VT [o]ption name'? DataValue for pair-field (k column)
s #string DataTypeCode for pair-field (v column)
string #OptionName (DataValue for pair-field (v column))
s #string DataTypeCode for pair-field (k column)
v #'[v]ariable name'? DataValue for pair-field (k column)
s #string DataTypeCode for pair-field (v column)
string #VariableName (DataValue for pair-field (v column))
```

6.14 Set VT Option

ActionID 12. Sets the VirindiTank option OptionName to the result of evaluating Expression. (See also Section 6.13.) Heed the red note at the start of Section 4.

ActionType and ActionData:

```
i #integer DataTypeCode for ActionType (or K) pair-field
12 #ActionID DataValue for ActionType (or K) pair-field
```

```
TABLE #TABLE DataTypeCode for ActionData (or V) pair-field
2 #ColumnCount for this TABLE
k #Column1 Name
v #Column2 Name
n #Column1 (k) is not indexed
n #Column2 (v) is not indexed
2 #RecordCount
s #string DataTypeCode for pair-field (k column)
o #‘VT [o]ption name’? DataValue for pair-field (k column)
s #string DataTypeCode for pair-field (v column)
string #OptionName (DataValue for pair-field (v column))
s #string DataTypeCode for pair-field (k column)
v #‘expression [v]alue’? DataValue for pair-field (k column)
s #string DataTypeCode for pair-field (v column)
string #Expression (DataValue for pair-field (v column))
```

6.15 Create View

ActionID 13. Creates a Virindi View with the designated ViewName (handle), the layout of which is defined by XML contained inside the ByteArray, which is exactly ByteCount characters (bytes) in length, including any newline characters encountered. (See Virindi’s Meta Views web page.) The end of the ByteArray in ‘Create View’ is the only place in all of this formatting where a newline character is not inserted to separate it from the follow-on data; that data begins immediately, on the same line as the end of the byte array. Heed the red note at the start of Section 4.

ActionType and ActionData:

```
i #integer DataTypeCode for ActionType (or K) pair-field
13 #ActionID DataValue for ActionType (or K) pair-field
```

```
TABLE #TABLE DataTypeCode for ActionData (or V) pair-field
2 #ColumnCount for this TABLE
k #Column1 Name
v #Column2 Name
n #Column1 (k) is not indexed
n #Column2 (v) is not indexed
2 #RecordCount
s #string DataTypeCode for pair-field (k column)
n #‘view [n]ame’? DataValue for pair-field (k column)
s #string DataTypeCode for pair-field (v column)
string #ViewName (DataValue for pair-field (v column))
s #string DataTypeCode for pair-field (k column)
x #‘[x]ml’? DataValue for pair-field (k column)
ba #ByteArray DataTypeCode for ActionData (or V) pair-field
integer #ByteCount (length prefix for ByteArray)
ByteArray #run of ByteCount chars, incl. any newlines #DataValue for ActionData (or V) pair-field
```

6.16 Destroy View

ActionID 14. Destroys the ViewName Virindi View. Heed the red note at the start of Section 4.

ActionType and ActionData:

```
i #integer DataTypeCode for ActionType (or K) pair-field
14 #ActionID DataValue for ActionType (or K) pair-field
```

```
TABLE #TABLE DataTypeCode for ActionData (or V) pair-field
2 #ColumnCount for this TABLE
k #Column1 Name
v #Column2 Name
n #Column1 (k) is not indexed
n #Column2 (v) is not indexed
1 #RecordCount
s #string DataTypeCode for pair-field (k column)
n #'view [n]ame'? DataValue for pair-field (k column)
s #string DataTypeCode for pair-field (v column)
string #ViewName (DataValue for pair-field (v column))
```

6.17 Destroy All Views

ActionID 15. Destroys all Virindi Views for this meta. Heed the red note at the start of Section 4.

ActionType and ActionData:

```
i #integer DataTypeCode for ActionType (or K) pair-field
15 #ActionID DataValue for ActionType (or K) pair-field
```

```
TABLE #TABLE DataTypeCode for ActionData (or V) pair-field
2 #ColumnCount for this TABLE
k #Column1 Name
v #Column2 Name
n #Column1 (k) is not indexed
n #Column2 (v) is not indexed
0 #RecordCount
```

7 .nav Files and Embedding Nav Routes

All of the following nav route content exists inside the ByteArray portion of the 'Load Embedded Nav Route' Action, beginning at the start of the first line following the ByteCount. (See Section 6.6.) It is important to note that all this material is exactly identical to .nav file contents, except that the first two lines are absent in .nav files. (They begin on line 3, with "uTank2 NAV 1.2". See Section 7.1 and Section 7.2.) There is also a corner case for the 'Load Embedded Nav Route' Action, where only lines 1 and 2 are present: when no nav route has been imported at all (empty or otherwise).

There are two main structures for nav routes: 'follow' and the other three ('circular', 'linear', and 'once'). See Table 4 for a summary of all NavrouteTypeID codes, and Table 5 for a summary of all NavNodeTypeID codes.

7.1 Follow

NavrouteTypeID 3. The ‘Follow’ nav route type differs from the other three in that it contains no ‘nav nodes’ to speak of. It holds its follow-target data and immediately ends. This nav route type causes your character to try to follow the designated target as it moves around.

```
string #NavrouteNameInGame
integer #NavrouteNodeCount
uTank2 NAV 1.2
3 #NavrouteTypeID for Follow is 3
string #TargetName
integer #TargetWorldID
```

7.2 Circular, Linear, and Once

NavrouteTypeIDs 1, 2, and 4, respectively. All three of these nav route types share a common structure, with the key parameter being NavrouteNodeCount on line 5 since it states how many NavNodes are defined in the nav route (zero or more). All three of these nav route types evaluate the defined nav route sequentially, node after node, according to these rules: when Circular gets to the nav route end, it starts over at the beginning and keeps going (forever); when Linear gets to the end, it starts back up the node list in reverse, then goes forward again when it gets back to the start, etc. (forever); Once consumes nodes as it goes, removing them from the nav route until it hits the end, and the nav route is empty. Note that the VirindiTank Route interface calls nodes ‘Navigation Waypoints’.

```
1 string #NavrouteNameInGame
2 integer #NavrouteNodeCount (directly for meta code?)
3 uTank2 NAV 1.2
4 integer #NavrouteTypeID (1=Circular, 2=Linear, 4=Once)
5 integer #NavrouteNodeCount (for nav route reader code?)
6 [NavNodes] #multiple lines; iterated to define NavrouteNodeCount number of NavNodes
```

7.2.1 Point

NavNodeTypeID 0. Means ‘run to these coordinates’. These nodes may be added to a Route either by clicking the ‘Add’ button or by issuing the chat command `/vt addnavpt [coords]`, where `[coords]` are optional. If omitted, VirindiTank uses your character’s current coordinates (in 3D); if included, they must be valid coordinates, such as `/vt addnavpt 42.847263N,17.853783W`. (I haven’t examined what the z-coordinate ends up being in the latter case.)

```
0 #NavNodeTypeID
double #CharacterXCoordinate
double #CharacterYCoordinate
double #CharacterZCoordinate
0
```

7.2.2 Portal (deprecated)

NavNodeTypeID 1. Means 'use this portal'. This nav route node type is deprecated and no longer available in the VirindiTank Route interface. (I haven't tested if PortalWorldID is forced to be ObjectClass=14 (Portal).) It still exists in some old .met and .nav files, and `metaf` can easily create them if so desired.

```
1 #NavNodeTypeID
double #CharacterXCoordinate
double #CharacterYCoordinate
double #CharacterZCoordinate
0
integer #PortalWorldID
```

7.2.3 Recall

NavNodeTypeID 2. Means 'cast this recall spell'. See Table 6 for all Recall SpellIDs I know to be valid in this node type.

```
2 #NavNodeTypeID
double #CharacterXCoordinate
double #CharacterYCoordinate
double #CharacterZCoordinate
0
integer #SpellID
```

7.2.4 Pause

NavNodeTypeID 3. Means 'wait this many seconds'.

```
3 #NavNodeTypeID
double #CharacterXCoordinate
double #CharacterYCoordinate
double #CharacterZCoordinate
0
double #SecondsToPause
```

7.2.5 Chat

NavNodeTypeID 4. Means 'send this text as chat'.

```
4 #NavNodeTypeID
double #CharacterXCoordinate
double #CharacterYCoordinate
double #CharacterZCoordinate
0
string #ChatToSend
```

7.2.6 Open Vendor

NavNodeTypeID 5. Means 'open this vendor's window'.

```
5 #NavNodeTypeID
double #CharacterXCoordinate
double #CharacterYCoordinate
double #CharacterZCoordinate
0
integer #VendorWorldID
string #VendorName
```

7.2.7 Portal/NPC

NavNodeTypeID 6. Means 'use this object'. I only know of three allowed values for ObjectClass: 37 (NPC), 14 (Portal), and 10 (Container, which is how hooked portal devices are classed).

```
6 #NavNodeTypeID
double #CharacterXCoordinate
double #CharacterYCoordinate
double #CharacterZCoordinate
0
string #ObjectName
integer #ObjectClass
True
double #ObjectXCoordinate
double #ObjectYCoordinate
double #ObjectZCoordinate
```

7.2.8 NPC Talk

NavNodeTypeID 7. Means 'talk to this NPC' (Non-Player Character).

```
7 #NavNodeTypeID
double #CharacterXCoordinate
double #CharacterYCoordinate
double #CharacterZCoordinate
0
string #NPCName
integer #ObjectClass
True
double #NPCXCoordinate
double #NPCYCoordinate
double #NPCZCoordinate
```

7.2.9 Checkpoint

NavNodeTypeID 8. Means 'before continuing, confirm with server that character location is these coordinates'. There is no Route graphical interface to add these nodes to a nav route; they must be added by issuing the chat command `/vt addnavcheckpoint [coords]`, where `[coords]` are optional. If omitted, VirindiTank uses your character's current coordinates (in 3D); if included, they must be valid coordinates, such as `/vt addnavcheckpoint 42.847263N,17.853783W`. (I haven't examined what the Z-coordinate ends up being in the latter case.)

```
8 #NavNodeTypeID
double #CharacterXCoordinate
double #CharacterYCoordinate
double #CharacterZCoordinate
0
```

7.2.10 Jump

NavNodeTypeID 9. Means 'jump character in designated direction with the indicated power (delay) and Shift-key-status'. There is no Route graphical interface to add these nodes to a nav route; they must be added by issuing the chat command `/vt addnavjump [heading] [shift] [milliseconds]`, where `[heading]` is the direction your character faces for the jump (in degrees (between 0 and 360), with North being 0, and going around clockwise), `[shift]` being either `True` or `False` to designate if the Shift key should be held while performing the jump, and `[milliseconds]` ranges from 0 to 1000 and indicates the 'power' of the jump in terms of how long the Space key should be held before release while performing the jump.

Example: `/vt addnavjump 156.3 True 292.78`

```
9 #NavNodeTypeID
double #CharacterXCoordinate
double #CharacterYCoordinate
double #CharacterZCoordinate
0
double #HeadingInDegrees
boolean #HoldShift (either True or False)
double #DelayInMilliseconds
```

7.2.11 Other

NavNodeTypeID 99. Rumored to exist, though I never encountered it. As I recall, I heard this was defined in the VirindiTank source code, though I never confirmed this myself. Its format is unknown to me, though I'd speculate at minimum that it's akin to Point and Checkpoint.

A ID Code Tables

Table 2: ConditionID codes.

ID Code	Condition (VirindiTank)	Condition (`metaf`)	Details
-1	Unassigned (unavailable)	Unassigned (unavailable)	Section 5.1
0	Never	Never	Section 5.2
1	Always	Always	Section 5.3
2	All	All	Section 5.4
3	Any	Any	Section 5.5
4	Chat Message	ChatMatch	Section 5.6
5	Pack Slots <=	MainsSlotsLE	Section 5.7
6	Seconds in state >=	SecsInStateGE	Section 5.8
7	Navroute empty	NavEmpty	Section 5.9
8	Character Death	Death	Section 5.10
9	Any Vendor Open	VendorOpen	Section 5.11
10	Vendor Closed	VendorClosed	Section 5.12
11	Inventory Item Count <=	ItemCountLE	Section 5.13
12	Inventory Item Count >=	ItemCountGE	Section 5.14
13	Monster Name Count Within Distance	MobsInDist_Name	Section 5.15
14	Monster Priority Count Within Distance	MobsInDist_Priority	Section 5.16
15	Need to Buff	NeedToBuff	Section 5.17
16	No Monsters Within Distance	NoMobsInDist	Section 5.18
17	Landblock ==	BlockE	Section 5.19
18	Landcell ==	CellE	Section 5.20
19	Portalspace Entered	IntoPortal	Section 5.21
20	Portalspace Exited	ExitPortal	Section 5.22
21	Not	Not	Section 5.23
22	Seconds in state (P) >=	PSecsInStateGE	Section 5.24
23	Time Left On Spell >=	SecsOnSpellGE	Section 5.25
24	Burden Percent >=	BuPercentGE	Section 5.26
25	Dist any route pt >=	DistToRteGE	Section 5.27
26	Expression	Expr	Section 5.28
27	ClientDialogPopup (unavailable)	ClientDialogPopup (unavailable)	Section 5.29
28	Chat Message Capture	ChatCapture	Section 5.30

Table 3: ActionID codes.

ID Code	Action (VirindiTank)	Action (metaf)	Details
-1	Unassigned (unavailable)	Unassigned (unavailable)	Section 6.1
0	None	None	Section 6.2
1	Set Meta State	SetState	Section 6.3
2	Chat Command	Chat	Section 6.4
3	All	DoAll	Section 6.5
4	Load Embedded Nav Route	EmbedNav	Section 6.6
5	Call Meta State	CallState	Section 6.7
6	Return From Call	Return	Section 6.8
7	Expression Action	DoExpr	Section 6.9
8	Chat Expression	ChatExpr	Section 6.10
9	Set Watchdog	SetWatchdog	Section 6.11
10	Clear Watchdog	ClearWatchdog	Section 6.12
11	Get VT Option	GetOpt	Section 6.13
12	Set VT Option	SetOpt	Section 6.14
13	Create View	CreateView	Section 6.15
14	Destroy View	DestroyView	Section 6.16
15	Destroy All Views	DestroyAllViews	Section 6.17

Table 4: NavrouteTypeID codes.

ID Code	Type (VirindiTank)	Type (metaf)	Details
1	Circular	circular	Section 7.2
2	Linear	linear	Section 7.2
3	Follow	follow (and 'flw')	Section 7.1
4	Once	once	Section 7.2

Table 5: NavNodeTypeID codes.

ID Code	Type (VirindiTank)	Type (metaf)	Details
–	(Follow (pseudo-node))	flw	Table 4 and Section 7.1
0	Point	pnt	Section 7.2.1
1	Portal (deprecated)	prt (deprecated but supported)	Section 7.2.2
2	Recall	rcl	Section 7.2.3
3	Pause	pau	Section 7.2.4
4	Chat	cht	Section 7.2.5
5	Open Vendor	vnd	Section 7.2.6
6	Portal/NPC	ptl	Section 7.2.7
7	NPC Talk	npc	Section 7.2.8
8	Checkpoint	chk	Section 7.2.9
9	Jump	jmp	Section 7.2.10
99	Other (unavailable)	otr (unavailable)	Section 7.2.11

Table 6: Recall SpellIDs.

SpellID	**'SpellID' (VirindiTank)**	**'SpellID' (`metaf`)**	**Details**
48	Primary	Primary Portal Recall	Section 7.2.3
2647	Secondary	Secondary Portal Recall	Section 7.2.3
1635	LS	Lifestone Recall	Section 7.2.3
1636	LS Sending	Lifestone Sending	Section 7.2.3
2645	Portal	Portal Recall	Section 7.2.3
2931	Aphus	Recall Aphus Lassel	Section 7.2.3
2023	Sanctuary	Recall the Sanctuary	Section 7.2.3
2943	Caul	Recall the Singularity Caul	Section 7.2.3
3865	GW	Glenden Wood Recall	Section 7.2.3
2041	Aerlinthe	Aerlinthe Recall	Section 7.2.3
2813	Mt. Lethe	Mount Lethe Recall	Section 7.2.3
2941	Ulgrim's	Ulgrim's Recall	Section 7.2.3
4084	Bur	Bur Recall	Section 7.2.3
4198	PtOIA	Paradox-touched Olthoi Infested Area Recall	Section 7.2.3
4128	Graveyard	Call of the Mhoire Forge	Section 7.2.3
4213	Colosseum	Colosseum Recall	Section 7.2.3
5175	Fac. Hub	Facility Hub Recall	Section 7.2.3
5330	Gear K. Camp	Gear Knight Invasion Area Camp Recall	Section 7.2.3
5541	Neftet	Lost City of Neftet Recall	Section 7.2.3
4214	Candeth	Return to the Keep	Section 7.2.3
5175	FacHub	~~Facility Hub Recall~~ (omitted; repeat of above)	Section 7.2.3
6150	Rynthid	Rynthid Recall	Section 7.2.3
6321	VR Rocks	Viridian Rise Recall	Section 7.2.3
6322	VR Tree	Viridian Rise Great Tree Recall	Section 7.2.3
6325	Soc. CH	Celestial Hand Stronghold Recall	Section 7.2.3
6327	Soc. RB	Radiant Blood Stronghold Recall	Section 7.2.3
6326	Soc. EW	Eldrytch Web Stronghold Recall	Section 7.2.3