The 'itl1' Resource

Intended as a replacement for the 'INTL' (1) resource, the basic 'itl1' resource provides information on long date formats: the order of the date elements, which elements to include, the names of days and months, and how to abbreviate the names. Each installed script has one or more 'itl1' resources. The resource ID for each 'itl1' resource is in the script's resource number range. The default 'itl1' resource for a script is specified by the script's 'itlb' resource. The basic 'itl1' resource specifies

- long date format (including month and day names)
- region code for this particular 'itl1' resource

However, this basic format presents several limitations. First, it assumes that 7 day names and 12 month names are sufficient,which is not true for some calendars. For example, the traditional Jewish calendar can have 13 months. Second, it assumes that day and month names can be abbreviated by simply truncating them to a fixed length, but this not true in many languages.

With system software version 7.0, the 'itl1' resource may be optionally extended to provide additional information that solves these problems. The fields present in the old format have not been moved. The routines in the **International Utilities Package** that generate date strings use information in the 'itl1' extension if it is present.

> **Note:** The 'INTL' (1) resource is obsolete and will not be supported in future versions of the system software.

The old 'itl1' format was identical to the 'INTL' (1) format, which ended with a variable-length field intended to be used for code that altered the standard sorting behavior. This "local routine" field has been ignored since the introduction of the 'itl0', 'itl1', and 'itl2' resources because the code for changing the sorting behavior was moved to the 'itl2' resource. Consequently, in most existing 'itl1' resources, the local routine field merely contains a single RTS instruction (hexadecimal 0x4E75). The extended format is now indicated by the presence of the hexadecimal value 0xA89F as the first word in the local routine field; this is the unimplemented trap instruction, which could not have been the first word of any valid local routine. The new Rez template for the 'itl1' type can be used to perform a DeRez operation on old-format 'itl1' resources with 0x4E75 in this field as well as extended-format 'itl1' resources in which the extended data begins with the value 0xA89F.

The extended data provides the following additional information, which you can see as part of the Rez type definition for the 'itl1' resource in the code example which follows this list:

- A version number. The byte-length version number in the old part of the 'itl1' resource has been used for various special purposes over the years, so this field provides a *real* version number.
- A separate format code. This code is distinct from the version number. The current extended format has a format code of 0.
- A calendar code. Multiple calendars may be available on some systems, and it is necessary to identify the particular calendar for use with this

'itl1' resource. Constants for the various calendars are provided in **Script Manager Data**.

- A list of extra day names (extraDays). This format is for those calendars with more than 7 days.
- A list of extra month names (extraMonths). This format is for those calendars with more than 12 months.
- A list of abbreviated day names (abbrevDays).
- A list of abbreviated month names (abbrevMonths).
- A list of additional date separators (extraSeparators). When parsing date strings, the **String2Date** function permits the separators in this list to be used in addition to the date separators specified elsewhere in the 'itl0' and 'itl1' resources.

The code example below shows the new Rez type for the 'itl1' resource

Code example:International date and time information

```
type 'itl1' {

    //day names
    array [7] {
        pstring[15];                    //Sunday, Monday...
    };

    //month names
    array [12] {
        pstring[15];                    //January, February...
    };
    byte    dayName, none=255;          //suppressDay
    byte    dayMonYear, monDayYear = 255;       //longDate format
    byte    noDayLeadZero, dayLeadZero = 255;   //dayLeading0
    byte;                               //abbrLen
    string[4];                          //st0
    string[4];                          //st1
    string[4];                          //st2
    string[4];                          //st3
    string[4];                          //st4
    byte        Region;                 //region code
    byte;                               //version
    switch {
        case oldFormat:
            key hex integer = $4E75;    //old-format key
        case extFormat:
            key hex integer = $A89F;    //extended-format key
            hex integer;                //version
            hex integer;                //format
            integer;                    //calendar code
            //offset to & length of extraDays table
            unsigned longint = extraDays >> 3;
            unsigned longint = (endExtraDays - extraDays) >> 3;
```

```
            //offset to & length of extraMonths table
            unsigned longint = extraMonths >> 3;
            unsigned longint = (endExtraMonths - extraMonths) >> 3;
            //offset to & length of abbrevDays table
            unsigned longint = abbrevDays >> 3;
            unsigned longint = (endAbbrevDays - abbrevDays) >> 3;
            //offset to & length of abbrevMonths table
            unsigned longint = abbrevMonths >> 3;
            unsigned longint = (endAbbrevMonths - abbrevMonths) >> 3;
            //offset to & length of extraSeparators table
            unsigned longint = extraSeparators >> 3;
            unsigned longint = (endExtraSeparators - extraSeparators) >> 3;
extraDays:          //count and list of extra day names
            integer = $$CountOf(extraDaysArray);
            array extraDaysArray {
                pstring;
            };
endExtraDays:
extraMonths:        //count and list of extra month names
            integer = $$CountOf(extraMonthArray);
            array extraMonthArray {
                pstring;
            };
endExtraMonths:
abbrevDays:             //count and list of abbreviated day names
            integer = $$CountOf(abbrevDaysArray);
            array abbrevDaysArray {
                pstring;
            };
endAbbrevDays:
abbrevMonths:           //count and list of abbreviated month names
            integer = $$CountOf(abbrevMonthArray);
            array abbrevMonthArray {
                pstring;
            };
endAbbrevMonths:
extraSeparators:        //count and list of extra date separator names
            integer = $$CountOf(extraSeparatorsArray);
            array extraSeparatorsArray {
                pstring;
            };
endExtraSeparators:
        };
};
```