BINARY SPACE
RELIABLE SPACE SYSTEMS

Definition of the Mimics Description Language (MDL)

All information is subject to change without notice and does not represent a commitment on the part of BINARY SPACE.
Release 1.07 (January 2016)

Table of Contents

Table of Figures

Document Change Log

Issue	Revision	Date	Affected	Reason for change
1	1	May 2000	All	New document
1	2	September 2001	Chapter 2	Introduction of the `VOLATILE` keyword for non-initialized parameters
1	3	August 2004	Chapter 3	Changed parameter status identifiers
1	4	September 2005	Chapter 3.2.9.	New mimics object (Image)
1	5	August 2008	Chapter 3.1.	Added support for bandwidth measurement
1	6	May 2011	Chapter 3.1.	Added '`GetPastValueTime`' function
1	7	January 2016	Chapter 3.3.	Added satellite tracking, pass & interlink functions

In der Weid 3, CH-8122 Binz
Tel: +41 44 8877987, Fax: +41 44 8877989, Email: info@binary-space.com, Web: www.binary-space.com

1. Introduction

The ***Mimics Description Language*** (MDL) is the programming language of SatView™ developed to ease the description of the dynamic behavior of mimics displays. It is a macro extension to the common C++ language. The associated library provides access to all kind of information related to telemetry data and grants extensive control over mimics objects.

2. Syntax

The source code of a mimics display always consists of one or more procedures, each of them associated with a mimics object. The procedures are executed automatically whenever one of the parameters used inside is updated. It is also possible to control the update behavior programmatically.
A procedure complying with the following syntax needs to be implemented for every mimics object that needs some kind of animation:

```
MIMICS OBJECT 'Name'
[PARAMETERS P1{,Pj};]
[VOLATILE P1{,Pk};]
[AUTOTRIGGER | NOAUTOTRIGGER]
BEGIN
   MDL Code
END
```

Comments:
The above notation is in *Extended Backus Naur Formalism* (EBNF).

Name	Name of the mimics object.
$P_1,...P_j$	Parameters used to control the behavior of the mimics object. A parameter declaration P_n can be preceded by the **STATIC** keyword which has the effect that the mimics object procedure is not re-calculated even if the parameter P_n is updated.
$P_1,...P_k$	Parameters eventually used to control the behavior of the mimics object.

By using the **VOLATILE** keyword the subsequent enumerated parameters are not checked whether or not they are initialized (i.e. do have a value). Each of the specified parameters must be contained in the list declared with the **PARAMETERS** keyword and may have a random value. By default, the procedure is not executed whenever one of the parameters $P_1...P_j$ has no value. With the **VOLATILE** keyword specified, the calculation in the procedure is performed although one or more of the parameters $P_1...P_k$ have no (or a random) value.

In der Weid 3, CH-8122 Binz
Tel: +41 44 8877987, Fax: +41 44 8877989, Email: info@binary-space.com, Web: www.binary-space.com

The keywords **AUTOTRIGGER** or **NOAUTOTRIGGER** control the execution of the procedure. The first of it checks the parameters used in the procedure in order to decide whether an update should take place or not whereas the second one forces a recalculation at the occurrence of every telemetry unit.

3. Library

Various library functions supported by the MDL provide easy access to the telemetry data characteristics and allow full control over the various mimics objects. The library is grouped into functions related to the control of mimics objects and others providing support in the area of telemetry data processing.

3.1. Telemetry Functions

The following functions are supported:

Function	Description
CString **GetTMUnitTag(**void**)**	Returns the identifier of the telemetry unit that is currently processed. Note: When the Packet Telemetry Standard (CCSDS 102.0-B-2) is supported the function returns the name of the telemetry packet. For a telemetry format based standard, it results in a string with the syntax: 'FORMAT: *n*' where *n* is the frame number.
CTimeTag **GetTMUnitTime(**void**)**	Returns the time associated with the telemetry unit. Note: The time identifies the moment when the telemetry unit was received on ground including eventual corrections to compensate any delays caused by the ground segment.
UINT **GetTMUnitID(**void**)**	Returns the number of the telemetry unit. Note: When the Packet Telemetry Standard (CCSDS 102.0-B-2) is supported the function returns the *On Board Reference Time* (OBRT) of the telemetry packet. For a telemetry format based standard, it returns the number known as format counter.

BOOL **GetTMUnitData(**INT *nBytePos*, BYTE &*nValue***)**	Returns the value of a byte at the specified location `nBytePos` (>= 0) in the variable `nValue`. Note: The function returns TRUE if the specified location is valid, FALSE otherwise.
BOOL **GetTMUnitData(**INT *nBytePos*, INT *nBitPos*,INT *nLength*, ULONGLONG &*nValue***)**	Returns the value of data at the specified location `nBytePos` (>= 0), `nBitPos` (0 <= `nBitPos` < 8), `nLength` (1 <= `nLength` <= 64) in the variable `nValue`. Note: The function returns TRUE if the specified location is valid, FALSE otherwise.
WORD **GetTMUnitQuality(**void**)**	Returns the data quality indication of the telemetry unit. It may be a combination of one or more of the following values: **TMUNIT_DATAQUALITY_GOOD** **TMUNIT_DATAQUALITY_BAD** **TMUNIT_SEQUENCEQUALITY_GOOD** **TMUNIT_SEQUENCEQUALITY_BAD** **TMUNIT_TIMECORRELATION_GOOD** **TMUNIT_TIMECORRELATION_BAD** Note: The value **TMUNIT_DATAQUALITY_NONE** is returned in case of an error.
type-specifier **GetValue(***Parameter-Id***)**	Returns the current (calibrated) value of parameter `Parameter-Id`. Note: If the parameter occurs more than once within a telemetry unit, the function returns the value of the first occurrence.
type-specifier **GetValue(***Parameter-Id*, INT *nOccurrence***)**	Returns the current (calibrated) value of parameter `Parameter-Id` at the occurrence specified by `nOccurrence` (>= 0). Note: If an illegal occurrence number is specified the functions return 0.

In der Weid 3, CH-8122 Binz
Tel: +41 44 8877987, Fax: +41 44 8877989, Email: info@binary-space.com, Web: www.binary-space.com

type-specifier **GetRawValue(***Parameter-Id***)**	Returns the current raw value of parameter `Parameter-Id`. Note: If the parameter occurs more than once within a telemetry unit, the function returns the value of the first occurrence.
type-specifier **GetRawValue(***Parameter-Id***,** INT *nOccurrence***)**	Returns the current raw value of parameter `Parameter-Id` at the occurrence specified by `nOccurrence` (>= 0). Note: If an illegal occurrence number is specified the functions return 0.
CTimeTag **GetValueTime(***Parameter-Id***)**	Returns the time associated with the current value of parameter `Parameter-Id`. Note: If the parameter occurs more than once within a telemetry unit, the function returns the value of the first occurrence.
CTimeTag **GetValueTime(***Parameter-Id***,** INT *nOccurrence***)**	Returns the time associated with the current value of parameter `Parameter-Id` at the occurrence specified by `nOccurrence` (>= 0). Note: If an illegal occurrence number is specified the functions return 0.
type-specifier **GetPastValue(***Parameter-Id***,** INT *nSample***)**	Returns a past (calibrated) value of parameter `Parameter-Id`. The variable `nSample` specifies how many samples in the past the value should be from. Note: If a parameter occurs more than once within a telemetry unit, each occurrence is counted as a sample.
type-specifier **GetPastRawValue(***Parameter-Id***,** INT *nSample***)**	Returns a past raw value of parameter `Parameter-Id`. The variable `nSample` specifies how many samples in the past the value should be from. Note: If a parameter occurs more than once within a telemetry unit, each occurrence is counted as a sample.

CTimeTag **GetPastValueTime(***Parameter-Id*, INT *nSample*)	Returns the time associated with a past value of parameter `Parameter-Id`. The variable *nSample* specifies how many samples in the past the value should be from. Note: If a parameter occurs more than once within a telemetry unit, each occurrence is counted as a sample.
type-specifier **CalculateValueAverage(** *Parameter-Id*,INT *nSamples*)	Returns the raw average value of the last `nSamples` samples of parameter `Parameter-Id`. Note: If less than `nSamples` samples have been collected, the function returns a floating average of the samples already encountered.
UINT **GetStatus(***Parameter-Id*)	Returns the status of the parameter `Parameter-Id` which may be a combination of the following values: **TMPARAMETER_STATUS_GOOD** **TMPARAMETER_STATUS_BAD** **TMPARAMETER_STATUS_NOLIMIT** **TMPARAMETER_STATUS_SOFTLIMIT** **TMPARAMETER_STATUS_HARDLIMIT** **TMPARAMETER_STATUS_DELTALIMIT** **TMPARAMETER_STATUS_VALID** **TMPARAMETER_STATUS_INVALID** Note: The value **TMPARAMETER_STATUS_NONE** is returned if the parameter has no value.
UINT **GetStatus(***Parameter-Id*, INT *nOccurrence*)	Returns the status of the parameter `Parameter-Id` for the specified occurrence `nOccurrence`. See above for possible values returned by this function.

In der Weid 3, CH-8122 Binz
Tel: +41 44 8877987, Fax: +41 44 8877989, Email: info@binary-space.com, Web: www.binary-space.com

double **GetTotalTMBandwidth()**	Returns the total amount of bits per second protocol overhead for the telemetry unit). Note: A value of 'NAN' is returned when no bandwidth information is available. Use the macro **isnan**(double *f*) to check for that result.
double **GetAvailableTMBandwidth()**	Returns the currently unused bandwidth as a number between 0 and 1. Note: A value of 'NAN' is returned when no bandwidth information or measurement is available. Use the macro **isnan**(double *f*) to check for that result.
double **GetMaxDiagnosticTMBandwidth()**	Returns the maximum of bits per second currently available for diagnostic purposes, dumps or reports. Note: A value of 'NAN' is returned when no bandwidth information is available. Use the macro **isnan**(double *f*) to check for that result.
double **GetAvailableDiagnosticTMBandwidth()**	Returns the bandwidth currently available for diagnostic purposes, dumps or reports as a number between 0 and 1. Note: A value of 'NAN' is returned when no bandwidth information or measurement is available. Use the macro **isnan**(double *f*) to check for that result.
CTimeTag **GetLastTMBandwidthMeasurementTime()**	Returns the time of the last bandwidth measurement. Note: A time equal to 0 is returned when no bandwidth information or measurement is available.

3.2. Mimics Object Functions

The following functions are supported:

3.2.1. Common Object Functions

Function	Description
VOID **SetPosition(**CONST RECT &*rect***)**	Moves the mimics object to the specified position.
CRect **GetPosition(**void**)**	Returns the current position of the mimics object. Note: The returned position is always normalized i.e. it does not take into account any possible rotation.
VOID **FlipHorizontal(**void**)**	Flips the mimics object versus its vertical axis.
VOID **FlipVertical(**void**)**	Flips the mimics object versus its horizontal axis.
VOID **Rotate(**double *fAngle***)**	Rotates the mimics object `fAngle` degrees in the counter-clockwise direction.
VOID **Show(**void**)**	Shows the mimics object if it previously was hidden.
VOID **Hide(**void**)**	Hides the mimics object.
BOOL **IsVisible(**void**)**	Checks if the mimics object is currently visible.
BOOL **Blink(**INT *nInterval***)**	Starts blinking the mimics object with an interval of `nInterval` milliseconds. Note: Set `nInterval` to 0 in order to stop the blinking.

3.2.2. Line Functions

Function	Description
VOID **Solid(**void**)** VOID **Dash(**void**)** VOID **Dot(**void**)** VOID **DashDot(**void**)** VOID **DashDotDot(**void**)** BOOL **IsSolid(**void**)** BOOL **IsDashed(**void**)** BOOL **IsDotted(**void**)** BOOL **IsDashDotted(**void**)** BOOL **IsDashDotDotted(**void**)**	Changes the style of the line or checks for a certain style.
VOID **Cross(**BOOL *bEnable***)**	Adds or removes a crossing symbol to the line.
BOOL **SetCrossPt(**double *fPt***)** BOOL **GetCrossPt(**double **fPt***)	Positions the crossing symbol on the line or returns the position of it. Note: A value of 0.0 for `fPt` sets the crossing symbol at the left most position of the line, a value of 1.0 moves it to the right most position. The functions return FALSE if the crossing symbol is not enabled or if `fPt` < 0.0 or `fPt` > 1.0.
BOOL **IsCross(**void**)**	Checks if a crossing symbol is positioned on the line.
VOID **Arrow(**BOOL *bEnable***)** VOID **DoubleArrow(**BOOL *bEnable***)**	Adds or removes a (double) arrow symbol to a line.
BOOL **IsArrow(**void**)** BOOL **IsDoubleArrow(**void**)**	Checks if a (double) arrow symbol is positioned on the line.
VOID **SetColor(**COLORREF *nColor***)** COLORREF **GetColor(**void**)**	Changes or returns the color of the line.
VOID **SetThickness(**INT *nWidth***)** INT **GetThickness(**void**)**	Changes or returns the thickness (in pixels) of the line.

3.2.3. Arc Functions

Function	Description
VOID **SetBorderSize(**INT *nSize***)** INT **GetBorderSize()**	Changes or returns the size (in pixels) of the arc border.
VOID **SetBorderStyle(**INT *nStyle***)** INT **GetBorderStyle()**	Changes or returns the style of the arc border. Valid styles are: **PS_SOLID** **PS_DASH** **PS_DOT** **PS_DASHDOT** **PS_DASHDOTDOT**
VOID **SetBorderColor(**COLORREF *nColor***)** COLORREF **GetBorderColor(**void**)**	Changes or returns the color of the arc border.
VOID **SetInteriorColor(**COLORREF *nColor***)** COLORREF **GetInteriorColor(**void**)**	Changes or returns the interior color of the arc.
VOID **SetInteriorHatch(**INT *nHatch*, COLORREF *nColor***)** BOOL **GetInteriorHatch(**INT &*nHatch*, COLORREF &*nColor***)**	Changes or returns the interior hatch of the arc together with its color. Possible values for *nHatch* are: **HT_SOLID** **HT_HORIZONTAL** **HT_VERTICAL** **HT_BDIAGONAL** **HT_FDIAGONAL** **HT_CROSS** **HT_DIAGCROSS** **HT_LPOINTS** **HT_MPOINTS** **HT_HPOINTS**
VOID **SetRadials(**double *fRadial1*, double *fRadial2***)** VOID **GetRadials(**double &*fRadial1*, double &*fRadial2***)**	Changes or returns the radial limitations of the arc. The valid range is 0.0 <= *fRadial1*, *fRadial2* <= 2*PI.

3.2.4. Circle Functions

Function	Description
VOID **SetBorderSize(**INT *nSize***)** INT **GetBorderSize()**	Changes or returns the size (in pixels) of the circle border.
VOID **SetBorderStyle(**INT *nStyle***)** INT **GetBorderStyle()**	Changes or returns the style of the circle border. Valid styles are: **PS_SOLID** **PS_DASH** **PS_DOT** **PS_DASHDOT** **PS_DASHDOTDOT**
VOID **SetBorderColor(**COLORREF *nColor***)** COLORREF **GetBorderColor(**void**)**	Changes or returns the color of the circle border.
VOID **SetInteriorColor(**COLORREF *nColor***)** COLORREF **GetInteriorColor(**void**)**	Changes or returns the interior color of the circle.
VOID **SetInteriorHatch(**INT *nHatch*, COLORREF *nColor***)** BOOL **GetInteriorHatch(**INT &*nHatch*, COLORREF &*nColor***)**	Changes or returns the interior hatch of the circle together with its color. Possible values for *nHatch* are: **HT_SOLID** **HT_HORIZONTAL** **HT_VERTICAL** **HT_BDIAGONAL** **HT_FDIAGONAL** **HT_CROSS** **HT_DIAGCROSS** **HT_LPOINTS** **HT_MPOINTS** **HT_HPOINTS**

3.2.5. Rectangle Functions

Function	Description
VOID **SetBorderSize(**INT *nSize***)** INT **GetBorderSize()**	Changes or returns the size (in pixels) of the rectangle border.
VOID **SetBorderStyle(**INT *nStyle***)** INT **GetBorderStyle()**	Changes or returns the style of the rectangle border. Valid styles are: **PS_SOLID** **PS_DASH** **PS_DOT** **PS_DASHDOT** **PS_DASHDOTDOT**
VOID **SetBorderColor(**COLORREF *nColor***)** COLORREF **GetBorderColor(**void**)**	Changes or returns the color of the rectangle border.
VOID **SetInteriorColor(**COLORREF *nColor***)** COLORREF **GetInteriorColor(**void**)**	Changes or returns the interior color of the rectangle.
VOID **SetInteriorHatch(**INT *nHatch*, COLORREF *nColor***)** BOOL **GetInteriorHatch(**INT &*nHatch*, COLORREF &*nColor***)**	Changes or returns the interior hatch of the rectangle together with its color. Possible values for `nHatch` are: **HT_SOLID** **HT_HORIZONTAL** **HT_VERTICAL** **HT_BDIAGONAL** **HT_FDIAGONAL** **HT_CROSS** **HT_DIAGCROSS** **HT_LPOINTS** **HT_MPOINTS** **HT_HPOINTS**

3.2.6. Triangle Functions

Function	Description
VOID **SetBorderSize(**INT *nSize***)** INT **GetBorderSize()**	Changes or returns the size (in pixels) of the triangle border.
VOID **SetBorderStyle(**INT *nStyle***)** INT **GetBorderStyle()**	Changes or returns the style of the triangle border. Valid styles are: **PS_SOLID** **PS_DASH** **PS_DOT** **PS_DASHDOT** **PS_DASHDOTDOT**
VOID **SetBorderColor(**COLORREF *nColor***)** COLORREF **GetBorderColor(**void**)**	Changes or returns the color of the triangle border.
VOID **SetInteriorColor(**COLORREF *nColor***)** COLORREF **GetInteriorColor(**void**)**	Changes or returns the interior color of the triangle.
VOID **SetInteriorHatch(**INT *nHatch*, COLORREF *nColor***)** BOOL **GetInteriorHatch(**INT &*nHatch*, COLORREF &*nColor***)**	Changes or returns the interior hatch of the triangle together with its color. Possible values for `nHatch` are: **HT_SOLID** **HT_HORIZONTAL** **HT_VERTICAL** **HT_BDIAGONAL** **HT_FDIAGONAL** **HT_CROSS** **HT_DIAGCROSS** **HT_LPOINTS** **HT_MPOINTS** **HT_HPOINTS**
VOID **SetEdges(**double *x1*,double *y1*, double *x2*,double *y2*, double *x3*,double *y3***)** VOID **GetEdges(**double &*x1*,double &*y1*, double &*x2*,double &*y2*, double &*x3*,double &*y3***)**	Changes or returns the edge points of the triangle. Valid ranges are: 0.0 <= x_i, y_i <= 1.0. Note: (*x1*,*y1*)=(0.0,1.0) (*x2*,*y2*)=(1.0,1.0) (*x3*,*y3*)=(0.5,0,0) The above points draw a symmetrical triangle with the peak looking towards the top of the screen.

3.2.7. Switch Functions

Function	Description
VOID **SetColor(**COLORREF *nColor***)** COLORREF **GetColor(**void**)**	Changes or returns the color of the switch. Note: This function affects all parts of a switch.
VOID **SetFrameColor(**COLORREF *nColor***)** COLORREF **GetFrameColor(**void**)**	Changes or returns the color of the switch frame.
VOID **SetInteriorFrameColor(** COLORREF *nColor***)** COLORREF **GetInteriorFrameColor(**void**)**	Changes or returns the color inside the switch frame.
VOID **SetCenterColor(**COLORREF *nColor***)** COLORREF **GetCenterColor(**void**)**	Changes or returns the color of the switch center frame.
VOID **SetInteriorCenterColor(** COLORREF *nColor***)** COLORREF **GetInteriorCenterColor(**void**)**	Changes or returns the color inside the switch center frame.
VOID **SetBarColor(**COLORREF *nColor***)** COLORREF **GetBarColor(**void**)**	Changes or returns the color of the switch bar.
VOID **SetStubsColor(**COLORREF *nColor***)** COLORREF **GetStubsColor(**void**)**	Changes or returns the color of the switch stubs.
VOID **SetThickness(**INT *nWidth***)** INT **GetThickness(**void**)**	Changes or returns the thickness (in pixels) of the switch. Note: This function affects all parts of a switch.
VOID **SetFrameThickness(**INT *nWidth***)** INT **GetFrameThickness(**void**)**	Changes or returns the width (in pixels) of the switch frame.
VOID **SetCenterThickness(**INT *nWidth***)** INT **GetCenterThickness(**void**)**	Changes or returns the width (in pixels) of the switch center frame.
VOID **SetBarThickness(**INT *nWidth***)** INT **GetBarThickness(**void**)**	Changes or returns the width (in pixels) of the switch bar.
VOID **SetStubsThickness(**INT *nWidth***)** INT **GetStubsThickness(**void**)**	Changes or returns the width (in pixels) of the switch stubs.
VOID **Open(**void**)** BOOL **IsOpen(**void**)** BOOL **IsClosed(**void**)** VOID **Close(**void**)**	Opens or closes the switch and checks for either state.

VOID **Pos1**(void) BOOL **IsPos1**(void) VOID **Pos2**(void) BOOL **IsPos2**(void) VOID **Pos3**(void) BOOL **IsPos3**(void)	Moves the switch into one of the specified positions or checks for a certain one.
VOID **Broken**(void) BOOL **IsBroken**(void)	Shows the switch in a broken state. Note: A switch in a broken state has no toggle bar. One of the above functions must be used to return to a normal state.

3.2.8. Text Functions

Function	Description
VOID **SetText**(LPCTSTR *pszText*) CString **GetText**(void)	Changes or returns the text label.
VOID **SetMode**(INT *nMode*) INT **GetMode**(void)	Changes or returns the background mode. Valid values for *nMode* are: **TRANSPARENT** **OPAQUE**
VOID **SetColor**(COLORREF *nColor*) COLORREF **GetColor**(void)	Changes or returns the color of the text label.
VOID **SetBackgroundColor**(COLORREF *nColor*) COLORREF **GetBackgroundColor**(void)	Changes or returns the background color of the text label. Note: Specifying a background color is only useful when using the **OPAQUE** background mode.
BOOL **SetFont**(CONST LOGFONT **pFont*) BOOL **GetFont**(LOGFONT **pFont*)	Changes or returns the font of the text label.
VOID **AlignLeft**(void) VOID **AlignCenter**(void) VOID **AlignRight**(void) VOID **AlignVertical**(BOOL *bEnable*) BOOL **IsLeftAligned**(void) BOOL **IsCenterAligned**(void) BOOL **IsRightAligned**(void) BOOL **IsVerticalAligned**(void)	Aligns a text label to the left, center, right or centers it vertically and checks for a certain alignment.

VOID **LeftToRightReading**(void) VOID **TopToBottomReading**(void) BOOL **IsLeftToRightReading**(void) BOOL **IsTopToBottomReading**(void)	Draws the text label horizontally or vertically (one character below the other) and checks for a certain reading.
VOID **SetTabChars**(INT *nChars*) INT **GetTabChars**(void)	Changes or returns the number of characters between TABs.
VOID **WrapWords**(BOOL *bEnable*) BOOL **IsWrappingWords**(void)	Enables or disables the word wrapping mode and checks if this mode is on or not.
VOID **SingleLine**(BOOL *bEnable*) BOOL **IsSingleLine**(void)	Limits the text label to one line and checks if this limitation is enabled or not.

3.2.9. Image Functions

Function	Description
VOID **SetFileName**(LPCTSTR *pszFileName*) CString **GetFileName**(void) VOID **SetImageOrigin**(CONST POINT &*pt*) CPoint **GetImageOrigin**(void)	Changes or returns the file name of the associated image. Changes or returns the offset of the image.
VOID **SetImageSize**(CONST SIZE &*size*) CSize **GetImageSize**(void)	Changes or returns the current size (in percent) of the image. Note: A setting of '`size.cx` = 100' and '`size.cy` = 100' indicates the original size.
VOID **SetImageTransparency**(BYTE *nFactor*) BYTE **GetImageTransparency**(void)	Changes or returns the transparency of the image. Note: A factor of 255 means full opacity and 0 completely transparent (invisible).

3.3. Satellite Tracking, Pass & Interlink Functions

An extensive interface is provided by the MDL to support satellite tracking, location pass predictions as well as satellite interlink calculations.

Function	Description
BOOL **CalculateSpacecraftOrbit(** CSpacecraft **pSpacecraft*, CONST CTimeKey &*tTime*)	Calculates the orbit characteristics of the spacecraft specified by *pSpacecraft*. Note: The *pSpacecraft* argument must be initialized with the name and NORAD number of the spacecraft those orbit characteristics should be calculated. Consult the 'Data Types' table below for more information about the 'CSpacecraft' class.
BOOL **CalculateSpacecraftState(** CONST CSpacecraft **pSpacecraft*, CONST CTimeKey &*tTime*, CSpacecraftState &*cState*)	Calculates the position & velocity vector (relative to the Sun) of the spacecraft specified by *pSpacecraft*. Note: The *pSpacecraft* argument must be initialized with at least the name of the spacecraft those state vector should be calculated. The NORAD number must only be supplied for Earth-centric spacecraft. Consult the 'Data Types' table below for more information about the 'CSpacecraft' and 'CSpacecraftState' class.
BOOL **CalculateSpacecraftPasses(** CSpacecraftPasses &*pPasses*)	Calculates the pass periods over one or multiple locations for one or more spacecraft. Note: The argument *pPasses* must be initialized before calling this function (as demonstrated in the sample below).
BOOL **CalculateSpacecraftInterlinks(** CSpacecraftInterlinks &*pInterlinks*)	Calculates the interlink periods between two or three spacecraft. Note: The argument *pInterlinks* must be initialized before calling this function (as demonstrated in the sample below).

<table>
<tr>
<td>double CalculateSpacecraftOrbitLongitude(
LPCTSTR pszSpacecraft,UINT nNORADID,
CONST CTimeKey &tTime)</td>
<td>Returns the longitude of the specified spacecraft <pszSpacecraft,nNORADID> at the time tTime.
Note:
• This function is available for Earth-centric spacecraft only
• The parameter tTime must be within an interval of a few days from current real-time in order to guarantee a precise result
• The returned longitude will be between 0...360 degrees
<table><tr><th>Argument</th><th>Description</th></tr><tr><td>pszSpacecraft</td><td>The name of spacecraft.</td></tr><tr><td>nNORADID</td><td>The NORAD identifier of the specified spacecraft.</td></tr><tr><td>tTime</td><td>The time for which the longitude should be calculated.</td></tr></table></td>
</tr>
<tr>
<td>double CalculateSpacecraftOrbitLatitude(
LPCTSTR pszSpacecraft,UINT nNORADID,
CONST CTimeKey &tTime)</td>
<td>Returns the latitude of the specified spacecraft <pszSpacecraft,nNORADID> at the time tTime.
Note:
• This function is available for Earth-centric spacecraft only
• The parameter tTime must be within an interval of a few days from current real-time in order to guarantee a precise result
• The returned latitude will be between -90...90 degrees
<table><tr><th>Argument</th><th>Description</th></tr><tr><td>pszSpacecraft</td><td>The name of spacecraft.</td></tr><tr><td>nNORADID</td><td>The NORAD identifier of the specified spacecraft.</td></tr><tr><td>tTime</td><td>The time for which the latitude should be calculated.</td></tr></table></td>
</tr>
</table>

<table>
<tr>
<td>double CalculateSpacecraftOrbitAltitude(
LPCTSTR pszSpacecraft,UINT nNORADID,
CONST CTimeKey &tTime)</td>
<td>Returns the altitude of the specified spacecraft <pszSpacecraft,nNORADID> at the time tTime.
Note:
• This function is available for Earth-centric spacecraft only
• The parameter tTime must be within an interval of a few days from current real-time in order to guarantee a precise result
• The returned altitude will be > 0 km
<table><tr><th>Argument</th><th>Description</th></tr><tr><td>pszSpacecraft</td><td>The name of spacecraft.</td></tr><tr><td>nNORADID</td><td>The NORAD identifier of the specified spacecraft.</td></tr><tr><td>tTime</td><td>The time for which the altitude should be calculated.</td></tr></table></td>
</tr>
<tr>
<td>double CalculateSpacecraftOrbitVelocity(
LPCTSTR pszSpacecraft,UINT nNORADID,
CONST CTimeKey &tTime)</td>
<td>Returns the velocity of the specified spacecraft <pszSpacecraft,nNORADID> at the time tTime.
Note:
• This function is available for Earth-centric spacecraft only
• The parameter tTime must be within an interval of a few days from current real-time in order to guarantee a precise result
• The returned velocity will be > 0 km/s
<table><tr><th>Argument</th><th>Description</th></tr><tr><td>pszSpacecraft</td><td>The name of spacecraft.</td></tr><tr><td>nNORADID</td><td>The NORAD identifier of the specified spacecraft.</td></tr><tr><td>tTime</td><td>The time for which the velocity should be calculated.</td></tr></table></td>
</tr>
</table>

CSpacecraftPosition **CalculateSpacecraftPosition(** LPCTSTR *pszSpacecraft*,UINT *nNORADID*, CONST CTimeKey &*tTime*)	Returns the position (relative to the Sun) of the specified spacecraft <*pszSpacecraft*, *nNORADID*> at the time *tTime*. Note: • For Earth-centric spacecraft (*nNORADID* <> 0) the parameter *tTime* must be within an interval of a few days from current real-time in order to guarantee a precise result • The returned position will be returned in form of the class 'CSpacecraftPosition'; its members *m_x*, *m_y*, *m_z* contain the position coordinates in km **Argument** — **Description** *pszSpacecraft* — The name of spacecraft. *nNORADID* — The NORAD identifier of the specified spacecraft. *tTime* — The time for which the position (relative to the Sun) should be calculated.
CSpacecraftVelocity **CalculateSpacecraftVelocity(** LPCTSTR *pszSpacecraft*,UINT *nNORADID*, CONST CTimeKey &*tTime*)	Returns the velocity (relative to the Sun) of the specified spacecraft <*pszSpacecraft*, *nNORADID*> at the time *tTime*. Note: • For Earth-centric spacecraft (*nNORADID* <> 0) the parameter *tTime* must be within an interval of a few days from current real-time in order to guarantee a precise result • The returned velocity will be returned in form of the class 'CSpacecraftVelocity'; its members *m_x*, *m_y*, *m_z* contain the velocity coordinates in km/s **Argument** — **Description** *pszSpacecraft* — The name of spacecraft. *nNORADID* — The NORAD identifier of the specified spacecraft. *tTime* — The time for which the velocity (relative to the Sun) should be calculated.

TIMETAG **CalculateSpacecraftPassStartTime(**
LPCTSTR *pszSpacecraft*,UINT *nNORADID*,
LPCTSTR *pszLocation*,
double *fLocationLongitude*,
double *fLocationLatitude*,
double *fLocationAltitude*,
CONST CTimeKey &*tStartTime*,
CONST CTimeSpan &*tInterval*)

Returns the begin of the next pass over the location <*pszLocation*,*fLocationLongitude*, *fLocationLatitude*,*fLocationAltitude*> of the specified spacecraft <*pszSpacecraft*, *nNORADID*> after the time *tStartTime* and within the subsequent *tInterval* interval.

Note:

- This function is available for Earth-centric spacecraft only
- The parameter *tStartTime* must be within an interval of a few days from current real-time in order to guarantee a precise result

Argument	Description
pszSpacecraft	The name of spacecraft.
nNORADID	The NORAD identifier of the specified spacecraft.
pszLocation	The name of pass-over location.
fLocationLongitude	The longitude (deg) of the pass-over location.
fLocationLatitude	The latitude (deg) of the pass-over location.
fLocationAltitude	The altitude (km) of the pass-over location.
tStartTime	Specifies the start time to be used to calculate the next pass over the specified location.
tInterval	Specifies the interval to be used to calculate the next pass over the specified location.

In der Weid 3, CH-8122 Binz
Tel: +41 44 8877987, Fax: +41 44 8877989, Email: info@binary-space.com, Web: www.binary-space.com

TIMETAG **CalculateSpacecraftPassStopTime(**
LPCTSTR *pszSpacecraft*,UINT *nNORADID*,
LPCTSTR *pszLocation*,
double *fLocationLongitude*,
double *fLocationLatitude*,
double *fLocationAltitude*,
CONST CTimeKey &*tStartTime*,
CONST CTimeSpan &*tInterval*)

Returns the end of the next pass over the location <*pszLocation*,*fLocationLongitude*, *fLocationLatitude*,*fLocationAltitude*> of the specified spacecraft <*pszSpacecraft*, *nNORADID*> after the time *tStartTime* and within the subsequent *tInterval* interval.

Note:

- This function is available for Earth-centric spacecraft only
- The parameter *tStartTime* must be within an interval of a few days from current real-time in order to guarantee a precise result

Argument	Description
pszSpacecraft	The name of spacecraft.
nNORADID	The NORAD identifier of the specified spacecraft.
pszLocation	The name of pass-over location.
fLocationLongitude	The longitude (deg) of the pass-over location.
fLocationLatitude	The latitude (deg) of the pass-over location.
fLocationAltitude	The altitude (km) of the pass-over location.
tStartTime	Specifies the start time to be used to calculate the next pass over the specified location.
tInterval	Specifies the interval to be used to calculate the next pass over the specified location.

TIMETAG
CalculateSpacecraftInterlinkStartTime(
LPCTSTR *pszSpacecraftA*,UINT *nNORADIDA*,
LPCTSTR *pszSpacecraftB*,UINT *nNORADIDB*,
CONST CTimeKey &*tStartTime*,
CONST CTimeSpan &*tInterval*)

Returns the begin of the next interlink session between the spacecraft <*pszSpacecraftA*, *nNORADIDA*> and <*pszSpacecraftB*, *nNORADIDB*> after the time *tStartTime* and within the subsequent *tInterval* interval.

Note:

- This function is available for Earth-centric spacecraft only
- The parameter *tStarTime* must be within an interval of a few days from current real-time in order to guarantee a precise result

Argument	Description
pszSpacecraftA	The name of first spacecraft.
nNORADIDA	The NORAD identifier of the first spacecraft.
pszSpacecraftB	The name of second spacecraft.
nNORADIDB	The NORAD identifier of the second spacecraft.
tStartTime	Specifies the start time to be used to calculate the next interlink session.
tInterval	Specifies the interval to be used to calculate the next interlink session.

In der Weid 3, CH-8122 Binz
Tel: +41 44 8877987, Fax: +41 44 8877989, Email: info@binary-space.com, Web: www.binary-space.com

TIMETAG
CalculateSpacecraftInterlinkStopTime(
LPCTSTR *pszSpacecraftA*,UINT *nNORADIDA*,
LPCTSTR *pszSpacecraftB*,UINT *nNORADIDB*,
CONST CTimeKey &*tStartTime*,
CONST CTimeSpan &*tInterval*)

Returns the end of the next interlink session between the spacecraft <*pszSpacecraftA*, *nNORADIDA*> and <*pszSpacecraftB*, *nNORADIDB*> after the time *tStartTime* and within the subsequent *tInterval* interval.

Note:

- This function is available for Earth-centric spacecraft only
- The parameter *tStartTime* must be within an interval of a few days from current real-time in order to guarantee a precise result

Argument	Description
pszSpacecraftA	The name of first spacecraft.
nNORADIDA	The NORAD identifier of the first spacecraft.
pszSpacecraftB	The name of second spacecraft.
nNORADIDB	The NORAD identifier of the second spacecraft.
tStartTime	Specifies the start time to be used to calculate the next interlink session.
tInterval	Specifies the interval to be used to calculate the next interlink session.

TIMETAG
CalculateSpacecraftRelaidInterlinkStartTime(
LPCTSTR *pszSpacecraftA*,UINT *nNORADIDA*,
LPCTSTR *pszSpacecraftVia*,UINT *nNORADIDVia*,
LPCTSTR *pszSpacecraftB*,UINT *nNORADIDB*,
CONST CTimeKey &*tStartTime*,
CONST CTimeSpan &*tInterval*)

Returns the begin of the next interlink session between the spacecraft <*pszSpacecraftA*,*nNORADIDA*> and <*pszSpacecraftB*,*nNORADIDB*> via the relais <*pszSpacecraftVia*,*nNORADIDVia*> after the time *tStartTime* and within the subsequent *tInterval* interval.

Note:

- This function is available for Earth-centric spacecraft only
- The parameter *tStartTime* must be within an interval of a few days from current real-time in order to guarantee a precise result

Argument	Description
pszSpacecraftA	The name of first spacecraft.
nNORADIDA	The NORAD identifier of the first spacecraft.
pszSpacecraftVia	The name of relais spacecraft.
nNORADIDVia	The NORAD identifier of the relais spacecraft.
pszSpacecraftB	The name of second spacecraft.
nNORADIDB	The NORAD identifier of the second spacecraft.
tStartTime	Specifies the start time to be used to calculate the next interlink session.
tInterval	Specifies the interval to be used to calculate the next interlink session.

TIMETAG
CalculateSpacecraftRelaidInterlinkStopTime(
LPCTSTR *pszSpacecraftA*,UINT *nNORADIDA*,
LPCTSTR *pszSpacecraftVia*,UINT *nNORADIDVia*,
LPCTSTR *pszSpacecraftB*,UINT *nNORADIDB*,
CONST CTimeKey &*tStartTime*,
CONST CTimeSpan &*tInterval*)

Returns the end of the next interlink session between the spacecraft <*pszSpacecraftA*,*nNORADIDA*> and <*pszSpacecraftB*,*nNORADIDB*> via the relais <*pszSpacecraftVia*,*nNORADIDVia*> after the time *tStartTime* and within the subsequent *tInterval* interval.

Note:

- This function is available for Earth-centric spacecraft only
- The parameter *tStartTime* must be within an interval of a few days from current real-time in order to guarantee a precise result

Argument	Description
pszSpacecraftA	The name of first spacecraft.
nNORADIDA	The NORAD identifier of the first spacecraft.
pszSpacecraftVia	The name of relais spacecraft.
nNORADIDVia	The NORAD identifier of the relais spacecraft.
pszSpacecraftB	The name of second spacecraft.
nNORADIDB	The NORAD identifier of the second spacecraft.
tStartTime	Specifies the start time to be used to calculate the next interlink session.
tInterval	Specifies the interval to be used to calculate the next interlink session.

Note:
All satellite tracking, pass & interlink functions cannot be tested within the SatView™ Editor; they all return 'NAN' (for 'double' data types) and '0' (for 'TIMETAG' data types). When executed within the SatView™ Desktop, the satellite tracking sub-system must be enabled for these functions to return valid results. Furthermore, it must be ensured that access to the Internet is guaranteed.

Helper Functions:

Function	Description
CString **ConvertToText(**INT *nValue***)** CString **ConvertToText(**UINT *nValue***)** CString **ConvertToText(**double *fValue***)**	Converts a numerical value into a string. Note: The default number of significant digits for floating point numbers is 8.
CString **ConvertToText(**double *fValue*,INT *nDigits***)**	Converts a floating point value into a string with a maximum of `nDigits` significant digits.

Data Types:

Identifier	Description
type-specifier	Depending on the data type of the telemetry parameter P_i it is either an UINT, INT, double or CString.
parameter-tag	Tag of the telemetry parameter P_i.
CSpacecraft	A class representing a spacecraft (incl. its orbit characteristics). The following member functions are available: VOID **SetName(**LPCTSTR *pszName***)** CString **GetName()** CONST VOID **SetNumber(**UINT *nNumber***)** UINT **GetNumber()** CONST CTimeKey **GetOrbitPosition(**double &*fLongitude*,double &*fLatitude*, double &*fAltitude*,double &*fSpeed***)** CONST CTimeKey **GetTLEEpoch()** CONST double **GetOrbitInclination()** CONST double **GetOrbitEccentricity()** CONST double **GetOrbitRAAN()** CONST double **GetOrbitArgPerigee()** CONST double **GetOrbitBStar()** CONST double **GetOrbitDrag()** CONST double **GetOrbitMeanAnomaly()** CONST double **GetOrbitMajorAxis()** CONST double **GetOrbitMinorAxis()** CONST double **GetOrbitPerigee()** CONST double **GetOrbitApogee()** CONST double **GetOrbitMeanMotion()** CONST double **GetOrbitPeriod()** CONST double **GetLaunchTime()** CONST Note: Only the first four functions are available for non Earth-centric spacecraft.

In der Weid 3, CH-8122 Binz
Tel: +41 44 8877987, Fax: +41 44 8877989, Email: info@binary-space.com, Web: www.binary-space.com

CSpacecraftState	A class representing the position & velocity vector of a spacecraft (relative to the Sun for all non Earth-centric ones). The following member functions are available: CSpacecraftPosition **GetPosition()** CONST CSpacecraftVelocity **GetVelocity()** CONST
CSpacecraftPosition	A class representing the position of a spacecraft (relative to the Sun for all non Earth-centric ones). The following member properties are available: double **m_x** double **m_y** double **m_z**
CSpacecraftVelocity	A class representing the velocity of a spacecraft (relative to the Sun for all non Earth-centric ones). The following member properties are available: double **m_x** double **m_y** double **m_z**
CSpacecraftPasses	A class representing a collection of 'CSpacecraftPass' items. Consult the *Microsoft® Foundation Class* (MFC) documentation (see the 'CPtrArray' class).
CSpacecraftPass	A class representing the properties of a spacecraft pass over one or multiple locations. The following member functions are available: VOID **SetName(**LPCTSTR *pszName*) CString **GetName()** CONST VOID **SetSpacecraft(**LPCTSTR *pszName*,UINT *nNumber*) CString **GetSpacecraft(**UINT &*nNumber*) CONST CString **GetSpacecraft()** CONST BOOL **SetLocations(**CONST CSpacecraftPassLocations &*pLocations*) INT **GetLocations(**CSpacecraftPassLocations &*pLocations*) CONST VOID **SetTimeInterval(**CONST CTimeKey &*tStartTime*, CONST CTimeSpan &*tDuration*) BOOL **GetTimeInterval(**CTimeKey &*tStartTime*,CTimeSpan &*tDuration*) CONST VOID **Enable(**BOOL *bEnable*=TRUE) BOOL **IsEnabled()** CONST
CSpacecraftPassLocations	A class representing a collection of 'CSpacecraftPassLocation' items. Consult the *Microsoft® Foundation Class* (MFC) documentation (see the 'CPtrArray' class).

In der Weid 3, CH-8122 Binz
Tel: +41 44 8877987, Fax: +41 44 8877989, Email: info@binary-space.com, Web: www.binary-space.com

CSpacecraftPassLocation	A class representing the properties of a location. The following member functions are available: VOID **SetName(**LPCTSTR *pszName*) CString **GetName()** CONST VOID **SetLongitude(**double *fLongitude*) double **GetLongitude()** CONST VOID **SetLatitude(**double *fLatitude*) double **GetLatitude()** CONST VOID **SetAltitude(**double *fAltitude*) double **GetAltitude()** CONST INT **GetLinkPeriods(**CSpacecraftPassPeriods &*pPeriods*) CONST
CSpacecraftPassPeriods	A class representing a collection of 'CSpacecraftPassPeriod' items. Consult the *Microsoft® Foundation Class* (MFC) documentation (see the 'CPtrArray' class).
CSpacecraftPassPeriod	A class representing the properties of a spacecraft location pass-over period. The following member functions are available: BOOL **GetPeriod(**CTimeKey &*tStartTime*,CTimeKey &*tMaxTime*, CTimeKey &*tStopTime*) CONST CTimeSpan **GetPeriodDuration()** CONST CTimeKey **GetStartTime()** CONST CTimeKey **GetMaximumTime()** CONST CTimeKey **GetStopTime()** CONST VOID **GetDirection(**PassPhase *nPhase*,double &*fAzimuth*, double &*fElevation*) CONST Note: The argument *nPhase* can have one of the following values: 'Start' (=0), 'Maximum' (=1) or 'End' (=2).
CSpacecraftInterlinks	A class representing a collection of 'CSpacecraftInterlink' items. Consult the *Microsoft® Foundation Class* (MFC) documentation (see the 'CPtrArray' class).
CSpacecraftInterlink	A class representing the properties of a spacecraft interlink. The following member functions are available: VOID **SetName(**LPCTSTR *pszName*) CString **GetName()** CONST VOID **SetSpacecraftA(**LPCTSTR *pszName*,UINT *nNumber*) CString **GetSpacecraftA(**UINT &*nNumber*) CONST CString **GetSpacecraftA()** CONST VOID **SetSpacecraftVia(**LPCTSTR *pszName*,UINT *nNumber*) CString **GetSpacecraftVia(**UINT &*nNumber*) CONST CString **GetSpacecraftVia()** CONST VOID **SetSpacecraftB(**LPCTSTR *pszName*,UINT *nNumber*) CString **GetSpacecraftB(**UINT &*nNumber*) CONST

	CString **GetSpacecraftB()** CONST VOID **SetTimeInterval(**CONST CTimeKey &*tStartTime*, CONST CTimeSpan &*tDuration***)** BOOL **GetTimeInterval(**CTimeKey &*tStartTime*,CTimeSpan &*tDuration***)** CONST INT **GetLinkPeriods(**CSpacecraftInterlinkPeriods &*pPeriods***)** CONST VOID **Enable(**BOOL *bEnable*=TRUE**)** BOOL **IsEnabled()** CONST
CSpacecraftInterlinkPeriods	A class representing a collection of 'CSpacecraftInterlinkPeriod' items. Consult the *Microsoft® Foundation Class* (MFC) documentation (see the 'CPtrArray' class).
CSpacecraftInterlinkPeriod	A class representing the properties of a spacecraft interlink period. The following member functions are available: BOOL **GetPeriod(**CTimeKey &*tStartTime*,CTimeKey &*tStopTime***)** CONST CTimeSpan **GetPeriodDuration()** CONST CTimeKey **GetStartTime()** CONST CTimeKey **GetStopTime()** CONST VOID **GetDirection(**InterlinkOrigin *nOrigin*,InterlinkPhase *nPhase*, double &*fAzimuth*,double &*fElevation***)** CONST Note: The argument *nOrigin* can have one of the following values: 'SpacecraftA' (=0) or 'SpacecraftB' (=1). *nPhase* can be either 'Start' (=0) or 'End' (=1).
CString	A class representing a string. Consult the *Microsoft® Foundation Class* (MFC) documentation.
CTimeTag TIMETAG	A class representing an absolute time in microseconds since January 1, 1970. Consult the *Microsoft® Foundation Class* (MFC) documentation (see the 'CTime' class).
CTimeKey TIMEKEY	A class representing an absolute time in seconds since January 1, 1970. Consult the *Microsoft® Foundation Class* (MFC) documentation (see the 'CTime' class).
CTimeSpan	A class representing a time interval in seconds. Consult the *Microsoft® Foundation Class* (MFC) documentation.
CRect	A class representing a rectangle. Consult the *Microsoft® Foundation Class* (MFC) documentation.
CPoint	A class representing a 2-dimensional point. Consult the *Microsoft® Foundation Class* (MFC) documentation.
CSize	A class representing a 2-dimensional size. Consult the *Microsoft® Foundation Class* (MFC) documentation.

4. Samples

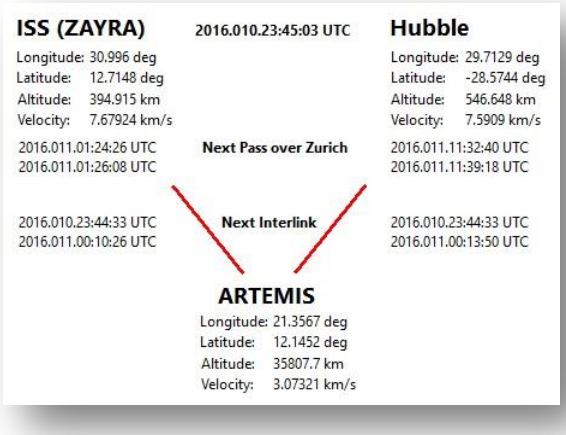
```
MIMICS OBJECT 'RKTS Switch'
PARAMETERS S351,S352;
BEGIN
  if (!GetRawValue(S351) && GetRawValue(S352)) Pos1();
  else if (GetRawValue(S351) && !GetRawValue(S352)) Pos3();
  else Broken();
END

MIMICS OBJECT 'OBS MODE Text'
PARAMETERS G201;
BEGIN
  SetText(G201);
END

MIMICS OBJECT 'BEACON A'
PARAMETERS G217;
BEGIN
  if (GetStatus(G217) & TMPARAMETER_STATUS_VALID)
  {
    if (!GetRawValue(G217)) SetInteriorColor(RGB(255,255,255));
    else SetInteriorColor(RGB(0,128,0));
  }
  else SetInteriorColor(RGB(255,255,255));
END

MIMICS OBJECT 'POINTER'
PARAMETERS S112;
BEGIN
  if (S112 > 10) Rotate(10.0);
  else Rotate(-10.0);
END
```

The following sample illustrates the use of the satellite tracking, pass & interlink functions:



In der Weid 3, CH-8122 Binz
Tel: +41 44 8877987, Fax: +41 44 8877989, Email: info@binary-space.com, Web: www.binary-space.com

```
MIMICS OBJECT 'Current Time (UTC)'
BEGIN
    if (GetTMUnitTime() > 0)
    {
      SetText(GetTMUnitTime().FormatGmt(TEXT("%Y.%j.%H:%M:%S UTC")));
    }
    else
    {
      SetText(TEXT("YYYY.ddd.HH:MM:SS UTC"));
    }
END

MIMICS OBJECT 'ISS Longitude'
BEGIN
    double  fSpeed;
    double  fAltitude;
    double  fLatitude;
    double  fLongitude;
    CSpacecraft  cSpacecraft;

    cSpacecraft.SetNumber(25544);
    cSpacecraft.SetName(TEXT("ISS (ZAYRA)"));
    if (CalculateSpacecraftOrbit(&cSpacecraft,GetTMUnitTime().GetTimeInSeconds()))
    {
      if (cSpacecraft.GetOrbitPosition(fLongitude,fLatitude,fAltitude,fSpeed) ==
GetTMUnitTime().GetTimeInSeconds())
      {
        SetText(ConvertToText(fLongitude,6)+TEXT(" deg"));
      }
      else
      {
        SetText(TEXT("xxx.xx deg"));
      }
    }
    else
    {
      SetText(TEXT("xxx.xx deg"));
    }
END

MIMICS OBJECT 'ISS Latitude'
BEGIN
    double  fSpeed;
    double  fAltitude;
    double  fLatitude;
    double  fLongitude;
    CSpacecraft  cSpacecraft;

    cSpacecraft.SetNumber(25544);
    cSpacecraft.SetName(TEXT("ISS (ZAYRA)"));
    if (CalculateSpacecraftOrbit(&cSpacecraft,GetTMUnitTime().GetTimeInSeconds()))
    {
      if (cSpacecraft.GetOrbitPosition(fLongitude,fLatitude,fAltitude,fSpeed) ==
GetTMUnitTime().GetTimeInSeconds())
```

In der Weid 3, CH-8122 Binz
Tel: +41 44 8877987, Fax: +41 44 8877989, Email: info@binary-space.com, Web: www.binary-space.com

```
      {
        SetText(ConvertToText(fLatitude,6)+TEXT(" deg"));
      }
      else
      {
        SetText(TEXT("xxx.xx deg"));
      }
    }
    else
    {
      SetText(TEXT("xxx.xx deg"));
    }
END

MIMICS OBJECT 'ISS Altitude'
BEGIN
    double  fSpeed;
    double  fAltitude;
    double  fLatitude;
    double  fLongitude;
    CSpacecraft  cSpacecraft;

    cSpacecraft.SetNumber(25544);
    cSpacecraft.SetName(TEXT("ISS (ZAYRA)"));
    if (CalculateSpacecraftOrbit(&cSpacecraft,GetTMUnitTime().GetTimeInSeconds()))
    {
      if (cSpacecraft.GetOrbitPosition(fLongitude,fLatitude,fAltitude,fSpeed) ==
GetTMUnitTime().GetTimeInSeconds())
      {
        SetText(ConvertToText(fAltitude,6)+TEXT(" km"));
      }
      else
      {
        SetText(TEXT("xxx.xxx km"));
      }
    }
    else
    {
      SetText(TEXT("xxx.xxx km"));
    }
END

MIMICS OBJECT 'ISS Velocity'
BEGIN
    double  fSpeed;
    double  fAltitude;
    double  fLatitude;
    double  fLongitude;
    CSpacecraft  cSpacecraft;

    cSpacecraft.SetNumber(25544);
    cSpacecraft.SetName(TEXT("ISS (ZAYRA)"));
    if (CalculateSpacecraftOrbit(&cSpacecraft,GetTMUnitTime().GetTimeInSeconds()))
    {
```

```
      if (cSpacecraft.GetOrbitPosition(fLongitude,fLatitude,fAltitude,fSpeed) ==
GetTMUnitTime().GetTimeInSeconds())
      {
        SetText(ConvertToText(fSpeed,6)+TEXT(" km/s"));
      }
      else
      {
        SetText(TEXT("xxx.xxx km/s"));
      }
    }
    else
    {
      SetText(TEXT("xxx.xxx km/s"));
    }
END

MIMICS OBJECT 'Hubble Longitude'
BEGIN
    double  fSpeed;
    double  fAltitude;
    double  fLatitude;
    double  fLongitude;
    CSpacecraft  cSpacecraft;

    cSpacecraft.SetNumber(20580);
    cSpacecraft.SetName(TEXT("Hubble"));
    if (CalculateSpacecraftOrbit(&cSpacecraft,GetTMUnitTime().GetTimeInSeconds()))
    {
      if (cSpacecraft.GetOrbitPosition(fLongitude,fLatitude,fAltitude,fSpeed) ==
GetTMUnitTime().GetTimeInSeconds())
      {
        SetText(ConvertToText(fLongitude,6)+TEXT(" deg"));
      }
      else
      {
        SetText(TEXT("xxx.xx deg"));
      }
    }
    else
    {
      SetText(TEXT("xxx.xx deg"));
    }
END

MIMICS OBJECT 'Hubble Latitude'
BEGIN
    double  fSpeed;
    double  fAltitude;
    double  fLatitude;
    double  fLongitude;
    CSpacecraft  cSpacecraft;

    cSpacecraft.SetNumber(20580);
    cSpacecraft.SetName(TEXT("Hubble"));
```

```
    if (CalculateSpacecraftOrbit(&cSpacecraft,GetTMUnitTime().GetTimeInSeconds()))
    {
      if (cSpacecraft.GetOrbitPosition(fLongitude,fLatitude,fAltitude,fSpeed) ==
GetTMUnitTime().GetTimeInSeconds())
      {
        SetText(ConvertToText(fLatitude,6)+TEXT(" deg"));
      }
      else
      {
        SetText(TEXT("xxx.xx deg"));
      }
    }
    else
    {
      SetText(TEXT("xxx.xx deg"));
    }
END

MIMICS OBJECT 'Hubble Altitude'
BEGIN
    double  fSpeed;
    double  fAltitude;
    double  fLatitude;
    double  fLongitude;
    CSpacecraft  cSpacecraft;

    cSpacecraft.SetNumber(20580);
    cSpacecraft.SetName(TEXT("Hubble"));
    if (CalculateSpacecraftOrbit(&cSpacecraft,GetTMUnitTime().GetTimeInSeconds()))
    {
      if (cSpacecraft.GetOrbitPosition(fLongitude,fLatitude,fAltitude,fSpeed) ==
GetTMUnitTime().GetTimeInSeconds())
      {
        SetText(ConvertToText(fAltitude,6)+TEXT(" km"));
      }
      else
      {
        SetText(TEXT("xxx.xxx km"));
      }
    }
    else
    {
      SetText(TEXT("xxx.xxx km"));
    }
END

MIMICS OBJECT 'Hubble Velocity'
BEGIN
    double  fSpeed;
    double  fAltitude;
    double  fLatitude;
    double  fLongitude;
    CSpacecraft  cSpacecraft;
```

In der Weid 3, CH-8122 Binz
Tel: +41 44 8877987, Fax: +41 44 8877989, Email: info@binary-space.com, Web: www.binary-space.com

```
    cSpacecraft.SetNumber(20580);
    cSpacecraft.SetName(TEXT("Hubble"));
    if (CalculateSpacecraftOrbit(&cSpacecraft,GetTMUnitTime().GetTimeInSeconds()))
    {
      if (cSpacecraft.GetOrbitPosition(fLongitude,fLatitude,fAltitude,fSpeed) ==
GetTMUnitTime().GetTimeInSeconds())
      {
        SetText(ConvertToText(fSpeed,6)+TEXT(" km/s"));
      }
      else
      {
        SetText(TEXT("xxx.xxx km/s"));
      }
    }
    else
    {
      SetText(TEXT("xxx.xxx km/s"));
    }
END

MIMICS OBJECT 'ARTEMIS Longitude'
BEGIN
    double  fSpeed;
    double  fAltitude;
    double  fLatitude;
    double  fLongitude;
    CSpacecraft  cSpacecraft;

    cSpacecraft.SetNumber(26863);
    cSpacecraft.SetName(TEXT("ARTEMIS"));
    if (CalculateSpacecraftOrbit(&cSpacecraft,GetTMUnitTime().GetTimeInSeconds()))
    {
      if (cSpacecraft.GetOrbitPosition(fLongitude,fLatitude,fAltitude,fSpeed) ==
GetTMUnitTime().GetTimeInSeconds())
      {
        SetText(ConvertToText(fLongitude,6)+TEXT(" deg"));
      }
      else
      {
        SetText(TEXT("xxx.xx deg"));
      }
    }
    else
    {
      SetText(TEXT("xxx.xx deg"));
    }
END

MIMICS OBJECT 'ARTEMIS Latitude'
BEGIN
    double  fSpeed;
    double  fAltitude;
    double  fLatitude;
    double  fLongitude;
```

```
    CSpacecraft  cSpacecraft;

    cSpacecraft.SetNumber(26863);
    cSpacecraft.SetName(TEXT("ARTEMIS"));
    if (CalculateSpacecraftOrbit(&cSpacecraft,GetTMUnitTime().GetTimeInSeconds()))
    {
      if (cSpacecraft.GetOrbitPosition(fLongitude,fLatitude,fAltitude,fSpeed) ==
GetTMUnitTime().GetTimeInSeconds())
      {
        SetText(ConvertToText(fLatitude,6)+TEXT(" deg"));
      }
      else
      {
        SetText(TEXT("xxx.xx deg"));
      }
    }
    else
    {
      SetText(TEXT("xxx.xx deg"));
    }
END

MIMICS OBJECT 'ARTEMIS Altitude'
BEGIN
    double  fSpeed;
    double  fAltitude;
    double  fLatitude;
    double  fLongitude;
    CSpacecraft  cSpacecraft;

    cSpacecraft.SetNumber(26863);
    cSpacecraft.SetName(TEXT("ARTEMIS"));
    if (CalculateSpacecraftOrbit(&cSpacecraft,GetTMUnitTime().GetTimeInSeconds()))
    {
      if (cSpacecraft.GetOrbitPosition(fLongitude,fLatitude,fAltitude,fSpeed) ==
GetTMUnitTime().GetTimeInSeconds())
      {
        SetText(ConvertToText(fAltitude,6)+TEXT(" km"));
      }
      else
      {
        SetText(TEXT("xxx.xxx km"));
      }
    }
    else
    {
      SetText(TEXT("xxx.xxx km"));
    }
END

MIMICS OBJECT 'ARTEMIS Velocity'
BEGIN
    double  fSpeed;
    double  fAltitude;
```

```
    double  fLatitude;
    double  fLongitude;
    CSpacecraft  cSpacecraft;

    cSpacecraft.SetNumber(26863);
    cSpacecraft.SetName(TEXT("ARTEMIS"));
    if (CalculateSpacecraftOrbit(&cSpacecraft,GetTMUnitTime().GetTimeInSeconds()))
    {
      if (cSpacecraft.GetOrbitPosition(fLongitude,fLatitude,fAltitude,fSpeed) ==
GetTMUnitTime().GetTimeInSeconds())
      {
        SetText(ConvertToText(fSpeed,6)+TEXT(" km/s"));
      }
      else
      {
        SetText(TEXT("xxx.xxx km/s"));
      }
    }
    else
    {
      SetText(TEXT("xxx.xxx km/s"));
    }
END

MIMICS OBJECT 'ISS Pass Start Time'
BEGIN
    CSpacecraftPass  *pPass;
    CSpacecraftPasses  pPasses;
    CSpacecraftPassPeriod  *pPassPeriod;
    CSpacecraftPassPeriods  pPassPeriods;
    CSpacecraftPassLocation  *pPassLocation;
    CSpacecraftPassLocations  pPassLocations;
    static CTimeKey  tPassStartTime=0;
    static CTimeKey  tPassStopTime=0;

    if (GetTMUnitTime().GetTimeInSeconds() > tPassStopTime.GetTime()) // Prevent
unnecessary calculations
    {
      if ((pPassLocation = new CSpacecraftPassLocation))
      {
        pPassLocation->SetName(TEXT("Zurich"));
        pPassLocation->SetLongitude(8.5500025);
        pPassLocation->SetLatitude(47.367347);
        pPassLocation->SetAltitude(0.425);
        if (pPassLocations.Add(pPassLocation) ≥ 0)
        {
        if ((pPass = new CSpacecraftPass))
        {
          pPass->SetName(TEXT("ISS Pass over Zurich"));
          pPass->SetSpacecraft(TEXT("ISS (ZAYRA)"),25544);
          pPass->SetLocations(pPassLocations);
          pPass->SetTimeInterval(GetTMUnitTime().GetTimeInSeconds(),86400);
          pPass->Enable(); // Enable the calculation of the pass period
          if (pPasses.Add(pPass) ≥ 0)
```

```
          {
            if (CalculateSpacecraftPasses(pPasses))
            {
              if ((pPass = pPasses.GetAt(0)))
              {
                if ((pPassLocation = (pPass->GetLocations(pPassLocations) > 0) ?
pPassLocations.GetAt(0):(CSpacecraftPassLocation *) NULL))
                {
                  if ((pPassPeriod = (pPassLocation->GetLinkPeriods(pPassPeriods) > 0) ?
pPassPeriods.GetAt(0):(CSpacecraftPassPeriod *) NULL))
                  {
                    SetText((tPassStartTime = pPassPeriod-
>GetStartTime()).FormatGmt(TEXT("%Y.%j.%H:%M:%S UTC")));
                    tPassStopTime = pPassPeriod->GetStopTime();
                  }
                  else
                  {
                    SetText(TEXT("YYYY.ddd.HH:MM:SS UTC"));
                  }
                }
                else
                {
                  SetText(TEXT("YYYY.ddd.HH:MM:SS UTC"));
                }
              }
              else
              {
                SetText(TEXT("YYYY.ddd.HH:MM:SS UTC"));
              }
            }
            else
            {
              SetText(TEXT("YYYY.ddd.HH:MM:SS UTC"));
              delete pPass;
            }
          }
          else
          {
            SetText(TEXT("YYYY.ddd.HH:MM:SS UTC"));
          }
        }
        else
        {
          SetText(TEXT("YYYY.ddd.HH:MM:SS UTC"));
          delete pPassLocation;
        }
      }
      else
      {
```

In der Weid 3, CH-8122 Binz
Tel: +41 44 8877987, Fax: +41 44 8877989, Email: info@binary-space.com, Web: www.binary-space.com

```
      SetText(TEXT("YYYY.ddd.HH:MM:SS UTC"));
    }
  }
END

MIMICS OBJECT 'ISS Pass End Time'
BEGIN
    CSpacecraftPass  *pPass;
    CSpacecraftPasses  pPasses;
    CSpacecraftPassPeriod  *pPassPeriod;
    CSpacecraftPassPeriods  pPassPeriods;
    CSpacecraftPassLocation  *pPassLocation;
    CSpacecraftPassLocations  pPassLocations;
    static CTimeKey  tPassStartTime=0;
    static CTimeKey  tPassStopTime=0;

    if (GetTMUnitTime().GetTimeInSeconds() > tPassStopTime.GetTime()) // Prevent
unnecessary calculations
    {
      if ((pPassLocation = new CSpacecraftPassLocation))
      {
        pPassLocation->SetName(TEXT("Zurich"));
        pPassLocation->SetLongitude(8.5500025);
        pPassLocation->SetLatitude(47.367347);
        pPassLocation->SetAltitude(0.425);
        if (pPassLocations.Add(pPassLocation) ≥ 0)
        {
        if ((pPass = new CSpacecraftPass))
        {
          pPass->SetName(TEXT("ISS Pass over Zurich"));
          pPass->SetSpacecraft(TEXT("ISS (ZAYRA)"),25544);
          pPass->SetLocations(pPassLocations);
          pPass->SetTimeInterval(GetTMUnitTime().GetTimeInSeconds(),86400);
          pPass->Enable(); // Enable the calculation of the pass period
          if (pPasses.Add(pPass) ≥ 0)
          {
            if (CalculateSpacecraftPasses(pPasses))
            {
              if ((pPass = pPasses.GetAt(0)))
              {
                if ((pPassLocation = (pPass->GetLocations(pPassLocations) > 0) ?
pPassLocations.GetAt(0):(CSpacecraftPassLocation *) NULL))
                {
                  if ((pPassPeriod = (pPassLocation->GetLinkPeriods(pPassPeriods) > 0) ?
pPassPeriods.GetAt(0):(CSpacecraftPassPeriod *) NULL))
                  {
                    tPassStartTime = pPassPeriod->GetStartTime();
                    SetText((tPassStopTime = pPassPeriod-
>GetStopTime()).FormatGmt(TEXT("%Y.%j.%H:%M:%S UTC")));
                  }
                  else
                  {
                    SetText(TEXT("YYYY.ddd.HH:MM:SS UTC"));
                  }
```

In der Weid 3, CH-8122 Binz
Tel: +41 44 8877987, Fax: +41 44 8877989, Email: info@binary-space.com, Web: www.binary-space.com

```
                }
                else
                {
                  SetText(TEXT("YYYY.ddd.HH:MM:SS UTC"));
                }
              }
              else
              {
                SetText(TEXT("YYYY.ddd.HH:MM:SS UTC"));
              }
            }
            else
            {
              SetText(TEXT("YYYY.ddd.HH:MM:SS UTC"));
            }
          }
          else
          {
            SetText(TEXT("YYYY.ddd.HH:MM:SS UTC"));
            delete pPass;
          }
        }
        else
        {
          SetText(TEXT("YYYY.ddd.HH:MM:SS UTC"));
        }
      }
      else
      {
        SetText(TEXT("YYYY.ddd.HH:MM:SS UTC"));
        delete pPassLocation;
      }
    }
    else
    {
      SetText(TEXT("YYYY.ddd.HH:MM:SS UTC"));
    }
  }
END

MIMICS OBJECT 'Hubble Pass Start Time'
BEGIN
    CSpacecraftPass  *pPass;
    CSpacecraftPasses  pPasses;
    CSpacecraftPassPeriod  *pPassPeriod;
    CSpacecraftPassPeriods  pPassPeriods;
    CSpacecraftPassLocation  *pPassLocation;
    CSpacecraftPassLocations  pPassLocations;
    static CTimeKey  tPassStartTime=0;
    static CTimeKey  tPassStopTime=0;

    if (GetTMUnitTime().GetTimeInSeconds() > tPassStopTime.GetTime()) // Prevent
unnecessary calculations
    {
```

In der Weid 3, CH-8122 Binz
Tel: +41 44 8877987, Fax: +41 44 8877989, Email: info@binary-space.com, Web: www.binary-space.com

```
      if ((pPassLocation = new CSpacecraftPassLocation))
      {
        pPassLocation->SetName(TEXT("Zurich"));
        pPassLocation->SetLongitude(8.5500025);
        pPassLocation->SetLatitude(47.367347);
        pPassLocation->SetAltitude(0.425);
        if (pPassLocations.Add(pPassLocation) ≥ 0)
        {
        if ((pPass = new CSpacecraftPass))
        {
          pPass->SetName(TEXT("ISS Pass over Zurich"));
          pPass->SetSpacecraft(TEXT("Hubble"),20580);
          pPass->SetLocations(pPassLocations);
          pPass->SetTimeInterval(GetTMUnitTime().GetTimeInSeconds(),86400);
          pPass->Enable(); // Enable the calculation of the pass period
          if (pPasses.Add(pPass) ≥ 0)
          {
            if (CalculateSpacecraftPasses(pPasses))
            {
              if ((pPass = pPasses.GetAt(0)))
              {
                if ((pPassLocation = (pPass->GetLocations(pPassLocations) > 0) ?
pPassLocations.GetAt(0):(CSpacecraftPassLocation *) NULL))
                {
                  if ((pPassPeriod = (pPassLocation->GetLinkPeriods(pPassPeriods) > 0) ?
pPassPeriods.GetAt(0):(CSpacecraftPassPeriod *) NULL))
                  {
                    SetText((tPassStartTime = pPassPeriod-
>GetStartTime()).FormatGmt(TEXT("%Y.%j.%H:%M:%S UTC")));
                    tPassStopTime = pPassPeriod->GetStopTime();
                  }
                  else
                  {
                    SetText(TEXT("YYYY.ddd.HH:MM:SS UTC"));
                  }
                }
                else
                {
                  SetText(TEXT("YYYY.ddd.HH:MM:SS UTC"));
                }
              }
              else
              {
                SetText(TEXT("YYYY.ddd.HH:MM:SS UTC"));
              }
            }
            else
            {
              SetText(TEXT("YYYY.ddd.HH:MM:SS UTC"));
            }
          }
          else
          {
            SetText(TEXT("YYYY.ddd.HH:MM:SS UTC"));
```

In der Weid 3, CH-8122 Binz
Tel: +41 44 8877987, Fax: +41 44 8877989, Email: info@binary-space.com, Web: www.binary-space.com

```
            delete pPass;
          }
        }
        else
        {
          SetText(TEXT("YYYY.ddd.HH:MM:SS UTC"));
        }
      }
      else
      {
        SetText(TEXT("YYYY.ddd.HH:MM:SS UTC"));
        delete pPassLocation;
      }
    }
    else
    {
      SetText(TEXT("YYYY.ddd.HH:MM:SS UTC"));
    }
  }
END

MIMICS OBJECT 'Hubble Pass End Time'
BEGIN
    CSpacecraftPass  *pPass;
    CSpacecraftPasses  pPasses;
    CSpacecraftPassPeriod  *pPassPeriod;
    CSpacecraftPassPeriods  pPassPeriods;
    CSpacecraftPassLocation  *pPassLocation;
    CSpacecraftPassLocations  pPassLocations;
    static CTimeKey  tPassStartTime=0;
    static CTimeKey  tPassStopTime=0;

    if (GetTMUnitTime().GetTimeInSeconds() > tPassStopTime.GetTime()) // Prevent
unnecessary calculations
    {
      if ((pPassLocation = new CSpacecraftPassLocation))
      {
        pPassLocation->SetName(TEXT("Zurich"));
        pPassLocation->SetLongitude(8.5500025);
        pPassLocation->SetLatitude(47.367347);
        pPassLocation->SetAltitude(0.425);
        if (pPassLocations.Add(pPassLocation) ≥ 0)
        {
        if ((pPass = new CSpacecraftPass))
        {
          pPass->SetName(TEXT("ISS Pass over Zurich"));
          pPass->SetSpacecraft(TEXT("Hubble"),20580);
          pPass->SetLocations(pPassLocations);
          pPass->SetTimeInterval(GetTMUnitTime().GetTimeInSeconds(),86400);
          pPass->Enable(); // Enable the calculation of the pass period
          if (pPasses.Add(pPass) ≥ 0)
          {
            if (CalculateSpacecraftPasses(pPasses))
            {
```

```
              if ((pPass = pPasses.GetAt(0)))
              {
                if ((pPassLocation = (pPass->GetLocations(pPassLocations) > 0) ?
pPassLocations.GetAt(0):(CSpacecraftPassLocation *) NULL))
                {
                  if ((pPassPeriod = (pPassLocation->GetLinkPeriods(pPassPeriods) > 0) ?
pPassPeriods.GetAt(0):(CSpacecraftPassPeriod *) NULL))
                  {
                    tPassStartTime = pPassPeriod->GetStartTime();
                    SetText((tPassStopTime = pPassPeriod-
>GetStopTime()).FormatGmt(TEXT("%Y.%j.%H:%M:%S UTC")));
                  }
                  else
                  {
                    SetText(TEXT("YYYY.ddd.HH:MM:SS UTC"));
                  }
                }
                else
                {
                  SetText(TEXT("YYYY.ddd.HH:MM:SS UTC"));
                }
              }
              else
              {
                SetText(TEXT("YYYY.ddd.HH:MM:SS UTC"));
              }
            }
            else
            {
              SetText(TEXT("YYYY.ddd.HH:MM:SS UTC"));
              delete pPass;
            }
          }
          else
          {
            SetText(TEXT("YYYY.ddd.HH:MM:SS UTC"));
          }
        }
        else
        {
          SetText(TEXT("YYYY.ddd.HH:MM:SS UTC"));
          delete pPassLocation;
        }
      }
      else
      {
        SetText(TEXT("YYYY.ddd.HH:MM:SS UTC"));
      }
    }
```

In der Weid 3, CH-8122 Binz
Tel: +41 44 8877987, Fax: +41 44 8877989, Email: info@binary-space.com, Web: www.binary-space.com

```
END

MIMICS OBJECT 'ISS-ARTEMIS Interlink Start Time'
BEGIN
    CSpacecraftInterlink  *pInterlink;
    CSpacecraftInterlinks  pInterlinks;
    CSpacecraftInterlinkPeriod  *pInterlinkPeriod;
    CSpacecraftInterlinkPeriods  pInterlinkPeriods;
    static CTimeKey  tInterlinkStartTime=0;
    static CTimeKey  tInterlinkStopTime=0;

    if (GetTMUnitTime().GetTimeInSeconds() > tInterlinkStopTime.GetTime()) // Prevent
unnecessary calculations     {
      if ((pInterlink = new CSpacecraftInterlink))
      {
        pInterlink->SetName(TEXT("ISS-ARTEMIS Interlink"));
        pInterlink->SetSpacecraftA(TEXT("ISS (ZAYRA)"),25544);
        pInterlink->SetSpacecraftB(TEXT("ARTEMIS"),26863);
        pInterlink->SetTimeInterval(GetTMUnitTime().GetTimeInSeconds(),86400);
        pInterlink->Enable(); // Enable the calculation of the interlink session
        if (pInterlinks.Add(pInterlink) ≥ 0)
        {
          if (CalculateSpacecraftInterlinks(pInterlinks))
          {
            if ((pInterlink = pInterlinks.GetAt(0)))
            {
              if ((pInterlinkPeriod = (pInterlink->GetLinkPeriods(pInterlinkPeriods) > 0)
? pInterlinkPeriods.GetAt(0):(CSpacecraftInterlinkPeriod *) NULL))
              {
                SetText((tInterlinkStartTime = pInterlinkPeriod-
>GetStartTime()).FormatGmt(TEXT("%Y.%j.%H:%M:%S UTC")));
                tInterlinkStopTime = pInterlinkPeriod->GetStopTime();
              }
              else
              {
                SetText(TEXT("YYYY.ddd.HH:MM:SS UTC"));
              }
            }
            else
            {
              SetText(TEXT("YYYY.ddd.HH:MM:SS UTC"));
            }
          }
          else
          {
            SetText(TEXT("YYYY.ddd.HH:MM:SS UTC"));
          }
        }
        else
        {
          SetText(TEXT("YYYY.ddd.HH:MM:SS UTC"));
          delete pInterlink;
        }
      }
```

In der Weid 3, CH-8122 Binz
Tel: +41 44 8877987, Fax: +41 44 8877989, Email: info@binary-space.com, Web: www.binary-space.com

```
      else
      {
        SetText(TEXT("YYYY.ddd.HH:MM:SS UTC"));
      }
    }
END

MIMICS OBJECT 'ISS-ARTEMIS Interlink End Time'
BEGIN
    CSpacecraftInterlink  *pInterlink;
    CSpacecraftInterlinks  pInterlinks;
    CSpacecraftInterlinkPeriod  *pInterlinkPeriod;
    CSpacecraftInterlinkPeriods  pInterlinkPeriods;
    static CTimeKey  tInterlinkStartTime=0;
    static CTimeKey  tInterlinkStopTime=0;

    if (GetTMUnitTime().GetTimeInSeconds() > tInterlinkStopTime.GetTime()) // Prevent
unnecessary calculations     {
      if ((pInterlink = new CSpacecraftInterlink))
      {
        pInterlink->SetName(TEXT("ISS-ARTEMIS Interlink"));
        pInterlink->SetSpacecraftA(TEXT("ISS (ZAYRA)"),25544);
        pInterlink->SetSpacecraftB(TEXT("ARTEMIS"),26863);
        pInterlink->SetTimeInterval(GetTMUnitTime().GetTimeInSeconds(),86400);
        pInterlink->Enable(); // Enable the calculation of the interlink session
        if (pInterlinks.Add(pInterlink) ≥ 0)
        {
          if (CalculateSpacecraftInterlinks(pInterlinks))
          {
            if ((pInterlink = pInterlinks.GetAt(0)))
            {
              if ((pInterlinkPeriod = (pInterlink->GetLinkPeriods(pInterlinkPeriods) > 0)
? pInterlinkPeriods.GetAt(0):(CSpacecraftInterlinkPeriod *) NULL))
              {
                tInterlinkStartTime = pInterlinkPeriod->GetStartTime();
                SetText((tInterlinkStopTime = pInterlinkPeriod-
>GetStopTime()).FormatGmt(TEXT("%Y.%j.%H:%M:%S UTC")));
              }
              else
              {
                SetText(TEXT("YYYY.ddd.HH:MM:SS UTC"));
              }
            }
            else
            {
              SetText(TEXT("YYYY.ddd.HH:MM:SS UTC"));
            }
          }
          else
          {
            SetText(TEXT("YYYY.ddd.HH:MM:SS UTC"));
          }
        }
        else
```

In der Weid 3, CH-8122 Binz
Tel: +41 44 8877987, Fax: +41 44 8877989, Email: info@binary-space.com, Web: www.binary-space.com

```
        {
          SetText(TEXT("YYYY.ddd.HH:MM:SS UTC"));
          delete pInterlink;
        }
      }
      else
      {
        SetText(TEXT("YYYY.ddd.HH:MM:SS UTC"));
      }
    }
END

MIMICS OBJECT 'ARTEMIS-Hubble Interlink Start Time'
BEGIN
    CSpacecraftInterlink  *pInterlink;
    CSpacecraftInterlinks  pInterlinks;
    CSpacecraftInterlinkPeriod  *pInterlinkPeriod;
    CSpacecraftInterlinkPeriods  pInterlinkPeriods;
    static CTimeKey  tInterlinkStartTime=0;
    static CTimeKey  tInterlinkStopTime=0;

    if (GetTMUnitTime().GetTimeInSeconds() > tInterlinkStopTime.GetTime()) // Prevent
unnecessary calculations     {
      if ((pInterlink = new CSpacecraftInterlink))
      {
        pInterlink->SetName(TEXT("ARTEMIS-Hubble Interlink"));
        pInterlink->SetSpacecraftA(TEXT("ARTEMIS"),26863);
        pInterlink->SetSpacecraftB(TEXT("Hubble"),20580);
        pInterlink->SetTimeInterval(GetTMUnitTime().GetTimeInSeconds(),86400);
        pInterlink->Enable(); // Enable the calculation of the interlink session
        if (pInterlinks.Add(pInterlink) ≥ 0)
        {
          if (CalculateSpacecraftInterlinks(pInterlinks))
          {
            if ((pInterlink = pInterlinks.GetAt(0)))
            {
              if ((pInterlinkPeriod = (pInterlink->GetLinkPeriods(pInterlinkPeriods) > 0)
? pInterlinkPeriods.GetAt(0):(CSpacecraftInterlinkPeriod *) NULL))
              {
                SetText((tInterlinkStartTime = pInterlinkPeriod-
>GetStartTime()).FormatGmt(TEXT("%Y.%j.%H:%M:%S UTC")));
                tInterlinkStopTime = pInterlinkPeriod->GetStopTime();
              }
              else
              {
                SetText(TEXT("YYYY.ddd.HH:MM:SS UTC"));
              }
            }
            else
            {
              SetText(TEXT("YYYY.ddd.HH:MM:SS UTC"));
            }
          }
          else
```

In der Weid 3, CH-8122 Binz
Tel: +41 44 8877987, Fax: +41 44 8877989, Email: info@binary-space.com, Web: www.binary-space.com

```
          {
            SetText(TEXT("YYYY.ddd.HH:MM:SS UTC"));
          }
        }
        else
        {
          SetText(TEXT("YYYY.ddd.HH:MM:SS UTC"));
          delete pInterlink;
        }
      }
      else
      {
        SetText(TEXT("YYYY.ddd.HH:MM:SS UTC"));
      }
    }
END

MIMICS OBJECT 'ARTEMIS-Hubble Interlink End Time'
BEGIN
    CSpacecraftInterlink  *pInterlink;
    CSpacecraftInterlinks  pInterlinks;
    CSpacecraftInterlinkPeriod  *pInterlinkPeriod;
    CSpacecraftInterlinkPeriods  pInterlinkPeriods;
    static CTimeKey  tInterlinkStartTime=0;
    static CTimeKey  tInterlinkStopTime=0;

    if (GetTMUnitTime().GetTimeInSeconds() > tInterlinkStopTime.GetTime()) // Prevent
unnecessary calculations     {
      if ((pInterlink = new CSpacecraftInterlink))
      {
        pInterlink->SetName(TEXT("ARTEMIS-Hubble Interlink"));
        pInterlink->SetSpacecraftA(TEXT("ARTEMIS"),26863);
        pInterlink->SetSpacecraftB(TEXT("Hubble"),20580);
        pInterlink->SetTimeInterval(GetTMUnitTime().GetTimeInSeconds(),86400);
        pInterlink->Enable(); // Enable the calculation of the interlink session
        if (pInterlinks.Add(pInterlink) ≥ 0)
        {
          if (CalculateSpacecraftInterlinks(pInterlinks))
          {
            if ((pInterlink = pInterlinks.GetAt(0)))
            {
              if ((pInterlinkPeriod = (pInterlink->GetLinkPeriods(pInterlinkPeriods) > 0)
? pInterlinkPeriods.GetAt(0):(CSpacecraftInterlinkPeriod *) NULL))
              {
                tInterlinkStartTime = pInterlinkPeriod->GetStartTime();
                SetText((tInterlinkStopTime = pInterlinkPeriod-
>GetStopTime()).FormatGmt(TEXT("%Y.%j.%H:%M:%S UTC")));
              }
              else
              {
                SetText(TEXT("YYYY.ddd.HH:MM:SS UTC"));
              }
            }
            else
```

```
            {
              SetText(TEXT("YYYY.ddd.HH:MM:SS UTC"));
            }
          }
          else
          {
            SetText(TEXT("YYYY.ddd.HH:MM:SS UTC"));
          }
        }
        else
        {
          SetText(TEXT("YYYY.ddd.HH:MM:SS UTC"));
          delete pInterlink;
        }
      }
      else
      {
        SetText(TEXT("YYYY.ddd.HH:MM:SS UTC"));
      }
    }
END

MIMICS OBJECT 'ISS-ARTEMIS Link Line'
BEGIN
    CSpacecraftInterlink  *pInterlink;
    CSpacecraftInterlinks  pInterlinks;
    CSpacecraftInterlinkPeriod  *pInterlinkPeriod;
    CSpacecraftInterlinkPeriods  pInterlinkPeriods;
    static CTimeKey  tInterlinkStartTime=0;
    static CTimeKey  tInterlinkStopTime=0;

    if (GetTMUnitTime().GetTimeInSeconds() > tInterlinkStopTime.GetTime()) // Prevent
unnecessary calculations
    {
      if ((pInterlink = new CSpacecraftInterlink))
      {
        pInterlink->SetName(TEXT("ISS-ARTEMIS Interlink"));
        pInterlink->SetSpacecraftA(TEXT("ISS (ZAYRA)"),25544);
        pInterlink->SetSpacecraftB(TEXT("ARTEMIS"),26863);
        pInterlink->SetTimeInterval(GetTMUnitTime().GetTimeInSeconds(),86400);
        pInterlink->Enable(); // Enable the calculation of the interlink session
        if (pInterlinks.Add(pInterlink) ≥ 0)
        {
          if (CalculateSpacecraftInterlinks(pInterlinks))
          {
            if ((pInterlink = pInterlinks.GetAt(0)))
            {
              if ((pInterlinkPeriod = (pInterlink->GetLinkPeriods(pInterlinkPeriods) > 0)
? pInterlinkPeriods.GetAt(0):(CSpacecraftInterlinkPeriod *) NULL))
              {
                tInterlinkStartTime = pInterlinkPeriod->GetStartTime();
                tInterlinkStopTime = pInterlinkPeriod->GetStopTime();
              }
              else
```

```
            {
              SetColor(RGB(192,192,192));
            }
          }
          else
          {
            SetColor(RGB(192,192,192));
          }
        }
        else
        {
          SetColor(RGB(192,192,192));
        }
      }
      else
      {
        SetColor(RGB(192,192,192));
        delete pInterlink;
      }
    }
    else
    {
      SetColor(RGB(192,192,192));
    }
  }
  SetColor((GetTMUnitTime().GetTimeInSeconds() ≥ tInterlinkStartTime.GetTime() && GetTM
UnitTime().GetTimeInSeconds() ≤ tInterlinkStopTime.GetTime() && tInterlinkStartTime > 0)
? RGB(255,0,0):RGB(192,192,192));
END

MIMICS OBJECT 'ARTEMIS-Hubble Link Line'
BEGIN
  CSpacecraftInterlink  *pInterlink;
  CSpacecraftInterlinks  pInterlinks;
  CSpacecraftInterlinkPeriod  *pInterlinkPeriod;
  CSpacecraftInterlinkPeriods  pInterlinkPeriods;
  static CTimeKey  tInterlinkStartTime=0;
  static CTimeKey  tInterlinkStopTime=0;

  if (GetTMUnitTime().GetTimeInSeconds() > tInterlinkStopTime.GetTime()) // Prevent
unnecessary calculations
  {
    if ((pInterlink = new CSpacecraftInterlink))
    {
      pInterlink->SetName(TEXT("ARTEMIS-Hubble Interlink"));
      pInterlink->SetSpacecraftA(TEXT("ARTEMIS"),26863);
      pInterlink->SetSpacecraftB(TEXT("Hubble"),20580);
      pInterlink->SetTimeInterval(GetTMUnitTime().GetTimeInSeconds(),86400);
      pInterlink->Enable(); // Enable the calculation of the interlink session
      if (pInterlinks.Add(pInterlink) ≥ 0)
      {
        if (CalculateSpacecraftInterlinks(pInterlinks))
        {
          if ((pInterlink = pInterlinks.GetAt(0)))
```

```
            {
              if ((pInterlinkPeriod = (pInterlink->GetLinkPeriods(pInterlinkPeriods) > 0)
? pInterlinkPeriods.GetAt(0):(CSpacecraftInterlinkPeriod *) NULL))
              {
                tInterlinkStartTime = pInterlinkPeriod->GetStartTime();
                tInterlinkStopTime = pInterlinkPeriod->GetStopTime();
              }
              else
              {
                SetColor(RGB(192,192,192));
              }
            }
            else
            {
              SetColor(RGB(192,192,192));
            }
          }
          else
          {
            SetColor(RGB(192,192,192));
          }
        }
        else
        {
          SetColor(RGB(192,192,192));
          delete pInterlink;
        }
      }
      else
      {
        SetColor(RGB(192,192,192));
      }
    }
    SetColor((GetTMUnitTime().GetTimeInSeconds() ≥ tInterlinkStartTime.GetTime() && GetTM
UnitTime().GetTimeInSeconds() ≤ tInterlinkStopTime.GetTime() && tInterlinkStartTime > 0)
? RGB(255,0,0):RGB(192,192,192));
END
```

A. Acceptance

This document has been read and accepted by ESA.

In der Weid 3, CH-8122 Binz
Tel: +41 44 8877987, Fax: +41 44 8877989, Email: info@binary-space.com, Web: www.binary-space.com