OLIVE SOFTWARE

ViewPoint 2.1
ORL Reference Guide

Version 1

October 2007

ViewPoint 2.1 ORL Reference Guide

Document Version 1

October 2007

Olive Software may change this manual, correct inaccurate current information, or change or improve the application at any time without prior notice. Some of these changes may be incorporated into new versions of this manual.

Olive Software Inc.
2953 Bunker Hill Lane
Suite 203
Santa Clara
95054

Tel: 408.200.1780
Fax: 408-200-1790

E-mail: info@olivesoftware.com

Table of Contents

Table of Contents **3**

1. Olive Recognition Language (ORL) SDK **20**

1.1 Preprocessing 20

1.2 Segmentation 20

1.3 Warehouse 20

1.4 Publication 20

2. How to Write ORL Scripts **20**

2.1 XMDTemplate.js 21

2.2 XMD4Include.js 22

2.2.1 Reserved Names of OlvPropertySet records 22

2.2.2 Reserved Flags of OlvPropertySet records 25

2.3 Predefined values and functions to be used in in XMD4Include.js scripts 26

2.3.1 Enumerations used by methods as return values or arguments 26

2.3.2 Functions toArray() and toSafeArray() 28

2.4 Standard scripts of the generic workflow 29

2.4.1 AP_FindHeadlines.js 29

2.4.2 AP_RevisePictureRelatedBlocks.js 29

2.4.3 AP_HierarchyRecognition.js 29

2.4.4 AP_AssignElementTypesToHeadings.js 30

2.4.5 AP_SetPageBasedEntities.js 30

2.4.6 AP_SetEntitiesFromTOCHeadings.js 30

2.5 ORL Objects Overview 30

2.5.1 General Objects 30

2.6 Basic Objects 31

2.6.1 Document-Oriented Objects 31

2.6.2 Functional Objects 31

3. ORL Hierarchy **32**

3.1 Sample Scripts 33

4. OlvTextStyleGallery **33**

4.1.1 Description: 33

4.1.2 Derived from: 33

4.1.3 Creation: 33

4.2 PROPERTIES AND METHODS 33

4.2.1 Properties Read/Write: 33

4.2.2 Properties Read only: 33

4.3 NumberOfTextStyles : integer 33

4.4 NumberOfCharacters : integer 33

4.5 NumberOfLetters : integer 33

4.6 NumberOfCapitals : integer 33

4.7 NumberOfDigits : integer 33

4.7.1 Example: 34

4.7.2 Methods: 34

4.7.3 GetTextStyleInfo 34

4.7.4 Description: 34

4.7.5 Prototype: 34

4.7.6 Parameters: 34
4.7.7 Return value: 34
4.7.8 Example: 34
4.7.9 GetPopularTextStyleInfo 35
4.7.10 Description: 35
4.7.11 Prototype: 35
4.7.12 Parameters: 35
4.7.13 Return value: 35
4.7.14 Example: 35

5. OlvPropertySet 35

5.1.1 Derived from: 36
5.1.2 Creation: 36
5.2 PROPERTIES AND METHODS 36
5.2.1 Properties Read/Write: 36
5.2.2 Properties Read only: 36
5.2.3 Methods: 36
5.3 SetProperty 36
5.3.1 Prototype: 36
5.3.2 Parameters: 37
5.3.3 Return value: 37
5.3.4 Example: 37
5.4 RemoveProperty 37
5.4.1 Prototype: 37
5.4.2 Parameters: 37
5.4.3 Return value: 37
5.4.4 Example: 37
5.5 ClearAllProperties 37
5.5.1 Prototype: 37
5.5.2 Parameters: 37
5.5.3 Return value: 38
5.5.4 Example: 38
5.6 CountProperties 38
5.6.1 Prototype: 38
5.6.2 Parameters: 38
5.6.3 Return value: 38
5.6.4 Example: 38
5.7 IsPropertyTemporary 38
5.7.1 Prototype: 39
5.7.2 Parameters: 39
5.7.3 Return value: 39
5.7.4 Example: 39
5.8 IsPropertyExists 39
5.8.1 Prototype: 39
5.8.2 Parameters: 39
5.8.3 Return value: 39
5.8.4 Example: 39
5.9 GetPropertyFlags 39
5.9.1 Prototype: 39
5.9.2 Parameters: 40
5.9.3 Return value: 40
5.9.4 Example: 40
5.10 GetPropertyValue 40
5.10.1 Prototype: 40
5.10.2 Parameters: 40
5.10.3 Return value: 40
5.10.4 Example: 40

5.11 SetMultiProperty 40
5.11.1 *PROTOTYPE:* *41*
5.11.2 *PARAMETERS:* *41*
5.11.3 *RETURN VALUE:* *41*
5.11.4 *EXAMPLE:* *41*
5.12 GetMultiPropertyValues 42
5.12.1 *PROTOTYPE:* *42*
5.12.2 *PARAMETERS:* *42*
5.12.3 *RETURN VALUE:* *42*
5.12.4 *EXAMPLE:* *42*
5.13 GetMultiPropertyFlags 42
5.13.1 *PROTOTYPE:* *42*
5.13.2 *PARAMETERS:* *42*
5.13.3 *RETURN VALUE:* *42*
5.13.4 *EXAMPLE:* *42*
5.14 AddMultiProperty 42
5.14.1 *DESCRIPTION:* *43*
5.14.2 *PROTOTYPE:* *43*
5.14.3 *PARAMETERS:* *43*
5.14.4 *RETURN VALUE:* *43*
5.14.5 *EXAMPLE:* *43*
5.15 AddPropertyFlags 43
5.15.1 *PROTOTYPE:* *43*
5.15.2 *PARAMETERS:* *43*
5.15.3 *RETURN VALUE:* *43*
5.15.4 *EXAMPLE:* *43*
5.16 ErasePropertyFlags 44
5.16.1 *PROTOTYPE:* *44*
5.16.2 *PARAMETERS:* *44*
5.16.3 *RETURN VALUE:* *44*
5.16.4 *EXAMPLE:* *44*

6. OlvDocCom 44

6.1.1 *DERIVED FROM:* *44*
6.1.2 *CREATION:* *44*
6.2 PROPERTIES AND METHODS 44
6.2.1 *PROPERTIES READ/WRITE:* *44*
6.2.2 *PROPERTIES READ ONLY:* *44*
6.3 NumberOfPages : Integer 44
6.3.1 *EXAMPLE:* *45*
6.4 SourceType : String 45
6.5 HasTOCPages : Boolean 45
6.6 IncludePath : String 45
6.7 DocPath : String 45
6.8 Methods: 45
6.9 GetExternalParameter 45
6.9.1 *PROTOTYPE:* *45*
6.9.2 *PARAMETERS:* *46*
6.9.3 *RETURN VALUE:* *46*
6.9.4 *EXAMPLE:* *46*
6.10 GetPage 46
6.10.1 *PROTOTYPE:* *46*
6.10.2 *PARAMETERS:* *46*
6.10.3 *RETURN VALUE:* *46*
6.10.4 *EXAMPLE:* *46*
6.11 GetIndexesOfTOCPages : VBArray 46

6.11.1 PROTOTYPE: 47
6.11.2 PARAMETERS: 47
6.11.3 RETURN VALUE: 47
6.11.4 EXAMPLE: 47
6.12 GetBlockID : Integer 47
6.12.1 PROTOTYPE: 47
6.12.2 PARAMETERS: 47
6.12.3 RETURN VALUE: 48
6.12.4 EXAMPLE: 48
6.13 GetBlockByID – not used by the script 48
6.13.1 PROTOTYPE: 48
6.13.2 PARAMETERS: 48
6.13.3 RETURN VALUE: 48
6.13.4 EXAMPLE: 48
6.14 RevisePictureRelatedBlocks 48
6.14.1 PROTOTYPE: 48
6.14.2 PARAMETERS: 48
6.14.3 RETURN VALUE: 48
6.14.4 EXAMPLE: 49

7. OlvPageCom 49

7.1.1 DERIVED FROM: 49
7.1.2 CREATION: 49
7.2 PROPERTIES AND METHODS 49
7.2.1 PROPERTIES READ/WRITE: 49
7.2.2 PROPERTIES READ ONLY: 49
7.3 PageNumber : Integer 49
7.3.1 EXAMPLE: 49
7.4 PageBox : OlvFRectCom 49
7.4.1 EXAMPLE: 49
7.5 ColumnsNumber: Integer 50
7.5.1 EXAMPLE: 50
7.6 TextBlocksNumber: Integer 50
7.6.1 EXAMPLE: 50
7.7 GraphBlocksNumber: Integer 50
7.7.1 EXAMPLE: 50
7.8 VSeparatorBlocksNumber: Integer 50
7.8.1 EXAMPLE: 50
7.9 HSeparatorBlocksNumber 50
7.9.1 EXAMPLE: 50
7.10 TableBlocksNumber 50
7.10.1 EXAMPLE: 51
7.10.2 METHODS: 51
7.11 GetTextBlock : OlvTextBlockCom 51
7.11.1 PROTOTYPE: 51
7.11.2 PARAMETERS: 51
7.11.3 RETURN VALUE: 51
7.11.4 EXAMPLE: 51
7.12 GetGraphBlock: OlvGraphBlockCom 51
7.12.1 PROTOTYPE: 51
7.12.2 PARAMETERS: 51
7.12.3 RETURN VALUE: 51
7.12.4 EXAMPLE: 52
7.13 GetHSeparatorBlock: OlvHSeparatorBlockCom 52
7.13.1 PROTOTYPE: 52
7.13.2 PARAMETERS: 52

7.13.3 RETURN VALUE: 52
7.13.4 EXAMPLE: 52
7.14 GetVSeparatorBlock: OlvVSeparatorBlockCom 52
7.14.1 PROTOTYPE: 52
7.14.2 PARAMETERS: 52
7.14.3 RETURN VALUE: 52
7.14.4 EXAMPLE: 52
7.15 GetTableBlock: OlvTableBlockCom 53
7.15.1 PROTOTYPE: 53
7.15.2 PARAMETERS: 53
7.15.3 RETURN VALUE: 53
7.15.4 EXAMPLE: 53
7.16 GetAllTextBlocks: VBArray 53
7.16.1 PROTOTYPE: 53
7.16.2 PARAMETERS: 53
7.16.3 RETURN VALUE: 53
7.16.4 EXAMPLE: 53
7.17 GetAllGraphBlocks: VBArray 54
7.17.1 PROTOTYPE: 54
7.17.2 PARAMETERS: 54
7.17.3 RETURN VALUE: 54
7.17.4 EXAMPLE: 54
7.18 GetAllHSeparatorBlocks: VBArray 54
7.18.1 PROTOTYPE: 54
7.18.2 PARAMETERS: 54
7.18.3 RETURN VALUE: 54
7.18.4 EXAMPLE: 54
7.19 GetColumnRange: OlvRangeCom 55
7.19.1 PROTOTYPE: 55
7.19.2 PARAMETERS: 55
7.19.3 RETURN VALUE: 55
7.19.4 EXAMPLE: 55
7.20 ComputeTextInBox: String 56
7.20.1 PROTOTYPE: 56
7.20.2 PARAMETERS: 56
7.20.3 RETURN VALUE: 56
7.20.4 EXAMPLE: 56
7.21 ComputeBiDiTextInBox: String 56
7.21.1 PROTOTYPE: 56
7.21.2 PARAMETERS: 56
7.21.3 RETURN VALUE: 56
7.21.4 EXAMPLE: 56

8. OlvTagCom 57

8.1.1 DERIVED FROM: 57
8.1.2 CREATION: 57
8.2 PROPERTIES AND METHODS 57
8.2.1 PROPERTIES READ/WRITE: 58
8.2.2 PROPERTIES READ ONLY: 58
8.3 StartLineIndex: Integer 58
8.4 EndLineIndex: Integer 58
8.5 StartQuadIndex: Integer 58
8.6 EndQuadIndex: Integer 58
8.6.1 EXAMPLE: 58
8.6.2 METHODS: 58
8.7 GetQuadBoxes: VBArray 58

8.7.1 PROTOTYPE: 58
8.7.2 PARAMETERS: 58
8.7.3 RETURN VALUE: 58
8.7.4 EXAMPLE: 59
8.7.5 USAGE EXAMPLE: 59

9. OlvBlockCom 59

9.1.1 DERIVED FROM: 60
9.1.2 CREATION: 60
9.2 PROPERTIES AND METHODS 60
9.2.1 PROPERTIES READ/WRITE: 60
9.2.2 PROPERTIES READ ONLY: 60
9.3 Type : String 60
9.4 TypeNumeric : integer 60
9.5 HasChildren : boolean 60
9.6 NumberOfChildren : integer 60
9.7 HasGParent : boolean 60
9.7.1 EXAMPLE: 61
9.7.2 METHODS: 61
9.8 GetChildren 61
9.8.1 DESCRIPTION: 61
9.8.2 PROTOTYPE: 61
9.8.3 PARAMETERS: 61
9.8.4 RETURN VALUE: 61
9.8.5 EXAMPLE: 61
9.9 GetGParent 61
9.9.1 DESCRIPTION: 61
9.9.2 PROTOTYPE: 61
9.9.3 PARAMETERS: 61
9.9.4 RETURN VALUE: 62
9.9.5 EXAMPLE: 62
9.10 Usage example: 62

10. OlvGraphBlockCom 64

10.1.1 DERIVED FROM: 64
10.1.2 CREATION: 65
10.2 PROPERTIES AND METHODS 65
10.2.1 PROPERTIES READ/WRITE: 65
10.2.2 PROPERTIES READ ONLY: 65
10.3 IsImage : boolean 65
10.3.1 EXAMPLE: 65
10.3.2 METHODS 65

11. OlvTableBlockCom 65

11.1.1 DERIVED FROM: 65
11.1.2 CREATION: 65
11.2 PROPERTIES AND METHODS 66
11.2.1 PROPERTIES READ/WRITE: 66
11.2.2 PROPERTIES READ ONLY: 66
11.2.3 METHODS: 66
11.2.4 USAGE EXAMPLE: 66

12. OlvHSeparatorBlockCom 66

12.1.1 DERIVED FROM: 67
12.1.2 CREATION: 67
12.2 PROPERTIES AND METHODS 67

12.2.1 PROPERTIES READ/WRITE: ... 67
12.2.2 PROPERTIES READ ONLY: ... 67
12.2.3 METHOD: ... 67
12.2.4 USAGE EXAMPLE: ... 67

13. OlvVSeparatorBlockCom ... 67

13.1.1 DERIVED FROM: ... 67
13.1.2 CREATION: ... 67
13.2 PROPERTIES AND METHODS ... 68
13.2.1 PROPERTIES READ/WRITE: ... 68
13.2.2 PROPERTIES READ ONLY: ... 68
13.2.3 METHOD: ... 68
13.2.4 USAGE EXAMPLE: ... 68

14. OlvTextLineCom ... 68

14.1.1 DERIVED FROM: ... 68
14.1.2 CREATION: ... 68
14.2 PROPERTIES AND METHODS ... 68
14.2.1 PROPERTIES READ/WRITE: ... 68
14.2.2 PROPERTIES READ ONLY: ... 68
14.3 Text : String ... 69
14.4 QuadsNumber : integer ... 69
14.4.1 EXAMPLE: ... 69
14.4.2 METHODS: ... 69
14.5 GetQuad ... 69
14.5.1 PROTOTYPE: ... 69
14.5.2 PARAMETERS: ... 69
14.5.3 RETURN VALUE: ... 69
14.5.4 EXAMPLE: ... 69
14.6 GetQuads ... 69
14.6.1 PROTOTYPE: ... 69
14.6.2 PARAMETERS: ... 70
14.6.3 RETURN VALUE: ... 70
14.6.4 EXAMPLE: ... 70
14.7 Usage Examples: ... 70

15. OlvQuadCom ... 71

15.1.1 DERIVED FROM: ... 71
15.1.2 CREATION: ... 71
15.2 PROPERTIES AND METHODS ... 72
15.2.1 PROPERTIES READ/WRITE: ... 72
15.2.2 PROPERTIES READ ONLY: ... 72
15.3 Text: string ... 72
15.4 FontFlags : number ... 72
15.5 FontName : string ... 72
15.6 FontSize : float ... 72
15.6.1 EXAMPLE: ... 72
15.6.2 METHODS: ... 72

16. OlvBlockFilterCom ... 73

16.1.1 DERIVED FROM: ... 73
16.1.2 CREATION: ... 73
16.2 PROPERTIES AND METHODS ... 73
16.2.1 PROPERTIES READ/WRITE: ... 73
16.3 FilterMatchCriteria: bollean ... 73
16.4 Box: OlvFRectCom ... 73

16.5 LocationRelations: integer ... 74
16.6 OrderRelations: integer ... 74
16.7 PageNo: integer ... 74
16.8 ColumnsInfo: OlvRangeCom ... 75
16.9 OrderRegime: integer ... 75
16.10 BlockType : integer ... 75
16.11 IgnoreRootBlocks: boolean ... 76
16.12 IgnoreChildBlocks: boolean ... 76
16.13 PagesRange: OlvRangeCom ... 76
16.14 PageStatus: integer (for future use) ... 76
16.15 PageLabel: string (for future use) ... 77
16.16 Inherited properties from OlvPropertySet ... 77
16.16.1 Properties Read only: ... *77*
16.16.2 Example: ... *77*
16.16.3 Methods: ... *77*
16.17 PassBlock ... 77
16.17.1 Prototype: ... *77*
16.17.2 Parameters: ... *77*
16.17.3 Return value: ... *77*
16.17.4 Example: ... *78*
16.18 PassPage ... 78
16.18.1 Prototype: ... *78*
16.18.2 Parameters: ... *78*
16.18.3 Return Value: ... *78*
16.18.4 Example: ... *78*
16.19 PassBlockNoPageFiltering ... 78
16.19.1 Prototype: ... *78*
16.19.2 Parameters: ... *78*
16.19.3 Return value: ... *78*
16.19.4 Example: ... *79*
16.20 FilterBlocks ... 79
16.20.1 Prototype: ... *79*
16.20.2 Parameters: ... *79*
16.20.3 Return Value: ... *79*
16.20.4 Example: ... *79*

17. OlvBlockSortingCom ... 80

17.1.1 Derived from: ... *80*
17.1.2 Creation: ... *80*
17.2 PROPERTIES AND METHODS ... 80
17.2.1 Properties Read/Write: ... *80*
17.2.2 Properties Read only: ... *80*
17.2.3 Methods: ... *80*
17.3 SortBlocks ... 80
17.3.1 Prototype: ... *80*
17.3.2 Parameters: ... *80*
17.3.3 Return value: ... *80*
17.3.4 Example: ... *81*

18. OlvFPointCom ... 81

18.1.1 Derived from: ... *81*
18.1.2 Creation: ... *81*
18.2 PROPERTIES AND METHODS ... 82
18.2.1 Properties Read/Write: ... *82*
18.2.2 Example: ... *82*
18.2.3 Properties Read only: ... *82*

18.2.4 METHODS: 82
18.3 Offset 82
18.3.1 PROTOTYPE: 82
18.3.2 PARAMETERS: 82
18.3.3 RETURN VALUE: 82
18.3.4 EXAMPLE: 83
18.4 IsEqual 83
18.4.1 PROTOTYPE: 83
18.4.2 PARAMETERS: 83
18.4.3 RETURN VALUE: 83
18.4.4 EXAMPLE: 83
18.5 Add 84
18.5.1 PROTOTYPE: 84
18.5.2 PARAMETERS: 84
18.5.3 RETURN VALUE: 84
18.5.4 EXAMPLE: 84
18.6 Sub 84
18.6.1 PROTOTYPE: 84
18.6.2 PARAMETERS: 84
18.6.3 RETURN VALUE: 84
18.6.4 EXAMPLE: 84
19. OlvFuzzyCompareCom 85
19.1.1 FOR EXAMPLE: 85
19.1.2 DERIVED FROM: 85
19.1.3 CREATION: 85
19.2 PROPERTIES AND METHODS 85
19.2.1 PROPERTIES READ/WRITE: 85
19.3 Delta: float 86
19.4 DeltaX : float 86
19.5 DeltaY : float 86
19.6 DeltasRelative : boolean 86
19.6.1 PROPERTIES READ ONLY: 86
19.6.2 EXAMPLE: 86
19.6.3 METHODS – COMPARING NUMBER TO NUMBER: 86
19.7 AreNumbersEqual 86
19.7.1 PROTOTYPE: 86
19.7.2 PARAMETERS: 87
19.7.3 RETURN VALUE: 87
19.7.4 EXAMPLE: 87
19.8 IsNumberSmallerThan 87
19.8.1 PROTOTYPE: 87
19.8.2 PARAMETERS: 88
19.8.3 RETURN VALUE: 88
19.8.4 EXAMPLE: 88
19.9 IsNumberGreaterThan 88
19.9.1 PROTOTYPE: 88
19.9.2 PARAMETERS: 89
19.9.3 RETURN VALUE: 89
19.9.4 EXAMPLE: 89
19.9.5 METHODS – COMPARING NUMBER TO RANGE: 89
19.10 IsNumberInsideRange 89
19.10.1 PROTOTYPE: 89
19.10.2 PARAMETERS: 89
19.10.3 RETURN VALUE: 90
19.10.4 EXAMPLE: 90

19.11 IsNumberOutsideRange ... 90
19.11.1 *Prototype:* ... 90
19.11.2 *Parameters:* ... 90
19.11.3 *Return value:* ... 90
19.11.4 *Example:* ... 90
19.12 IsNumberBeforeRange ... 91
19.12.1 *Prototype:* ... 91
19.12.2 *Parameters:* ... 91
19.12.3 *Return value:* ... 91
19.12.4 *Example:* ... 91
19.13 IsNumberAfterRange ... 91
19.13.1 *Prototype:* ... 91
19.13.2 *Parameters:* ... 91
19.13.3 *Return value:* ... 91
19.13.4 *Example:* ... 92
19.14 IsNumberAtRangeStar ... 92
19.14.1 *Prototype:* ... 92
19.14.2 *Parameters:* ... 92
19.14.3 *Return value:* ... 92
19.14.4 *Example:* ... 92
19.15 IsNumberAtRangeEnd ... 92
19.15.1 *Prototype:* ... 93
19.15.2 *Parameters:* ... 93
19.15.3 *Return value:* ... 93
19.15.4 *Example:* ... 93
19.15.5 *Methods – comparing Range to Range:* ... 93
19.16 IsRangeInsideOther ... 93
19.16.1 *Prototype:* ... 93
19.16.2 *Parameters:* ... 93
19.16.3 *Return value:* ... 93
19.16.4 *Example:* ... 94
19.17 IsRangeEncloseOther ... 94
19.17.1 *Prototype:* ... 94
19.17.2 *Parameters:* ... 94
19.17.3 *Return value:* ... 94
19.17.4 *Example:* ... 95
19.18 IsRangeIntersectOther ... 95
19.18.1 *Description:* ... 95
19.18.2 *Prototype:* ... 95
19.18.3 *Parameters:* ... 95
19.18.4 *Return value:* ... 95
19.18.5 *Example:* ... 95
19.19 IsRangeAfterOther ... 96
19.19.1 *Prototype:* ... 96
19.19.2 *Parameters:* ... 96
19.19.3 *Return value:* ... 96
19.19.4 *Example:* ... 96
19.20 IsRangeBeforeOther ... 97
19.20.1 *Prototype:* ... 97
19.20.2 *Parameters:* ... 97
19.20.3 *Return value:* ... 97
19.20.4 *Example:* ... 97
19.20.5 *Methods – comparing Point to Rectangle:* ... 97
19.21 IsPointBelowRect ... 97
19.21.1 *Prototype:* ... 98
19.21.2 *Parameters:* ... 98
19.21.3 *Return value:* ... 98

19.21.4 *EXAMPLE:* ... 98
19.22 IsPointRightToRect ... 98
19.22.1 *PROTOTYPE:* ... 98
19.22.2 *PARAMETERS:* ... 99
19.22.3 *RETURN VALUE:* ... 99
19.22.4 *EXAMPLE:* ... 99
19.23 IsPointRightLowerToRect ... 99
19.23.1 *PROTOTYPE:* ... 99
19.23.2 *PARAMETERS:* ... 99
19.23.3 *RETURN VALUE:* ... 100
19.23.4 *EXAMPLE:* ... 100
19.24 IsPointInsideRect ... 100
19.24.1 *PROTOTYPE:* ... 100
19.24.2 *PARAMETERS:* ... 100
19.24.3 *RETURN VALUE:* ... 100
19.24.4 *EXAMPLE:* ... 100
19.25 IsPointOutsideRect ... 101
19.25.1 *PROTOTYPE:* ... 101
19.25.2 *PARAMETERS:* ... 101
19.25.3 *RETURN VALUE:* ... 101
19.25.4 *EXAMPLE:* ... 101
19.26 IsPointCenterLowerToRect ... 102
19.26.1 *PROTOTYPE:* ... 102
19.26.2 *PARAMETERS:* ... 102
19.26.3 *RETURN VALUE:* ... 102
19.26.4 *EXAMPLE:* ... 102
19.27 IsPointRightCenterToRect ... 103
19.27.1 *PROTOTYPE:* ... 103
19.27.2 *PARAMETERS:* ... 103
19.27.3 *EXAMPLE:* ... 103
19.28 IsPointLeftToRect ... 104
19.28.1 *PROTOTYPE:* ... 104
19.28.2 *PARAMETERS:* ... 104
19.28.3 *RETURN VALUE:* ... 104
19.28.4 *EXAMPLE:* ... 104
19.29 IsPointAboveRect ... 104
19.29.1 *PROTOTYPE:* ... 104
19.29.2 *PARAMETERS:* ... 105
19.29.3 *RETURN VALUE:* ... 105
19.29.4 *EXAMPLE:* ... 105
19.30 IsPointLeftUpperToRect ... 105
19.30.1 *PROTOTYPE:* ... 105
19.30.2 *PARAMETERS:* ... 105
19.30.3 *RETURN VALUE:* ... 106
19.30.4 *EXAMPLE:* ... 106
19.31 IsPointCenterUpperToRect ... 106
19.31.1 *PROTOTYPE:* ... 106
19.31.2 *PARAMETERS:* ... 106
19.31.3 *RETURN VALUE:* ... 106
19.31.4 *EXAMPLE:* ... 107
19.32 IsPointLeftCenterToRect ... 107
19.32.1 *PROTOTYPE:* ... 107
19.32.2 *PARAMETERS:* ... 107
19.32.3 *RETURN VALUE:* ... 107
19.32.4 *EXAMPLE:* ... 107
19.33 IsPointRightUpperToRect ... 108
19.33.1 *PROTOTYPE:* ... 108

19.33.2 PARAMETERS: 108
19.33.3 RETURN VALUE: 108
19.33.4 EXAMPLE: 108
19.34 IsPointLeftLowerToRect 109
19.34.1 PROTOTYPE: 109
19.34.2 PARAMETERS: 109
19.34.3 RETURN VALUE: 109
19.34.4 EXAMPLE: 109
19.34.5 METHODS – COMPARING RECTANGLE TO RECTANGLE: 110
19.35 IsRectLeftToOther 110
19.35.1 PROTOTYPE: 110
19.35.2 PARAMETERS: 110
19.35.3 RETURN VALUE: 110
19.35.4 EXAMPLE: 110
19.36 IsRectRightToOther 111
19.36.1 PROTOTYPE: 111
19.36.2 PARAMETERS: 111
19.36.3 RETURN VALUE: 111
19.36.4 EXAMPLE: 111
19.37 IsRectAboveOther 112
19.37.1 PROTOTYPE: 112
19.37.2 PARAMETERS: 112
19.37.3 RETURN VALUE: 113
19.37.4 EXAMPLE: 113
19.38 IsRectBelowOther 113
19.38.1 PROTOTYPE: 113
19.38.2 PARAMETERS: 113
19.38.3 RETURN VALUE: 114
19.38.4 EXAMPLE: 114
19.39 IsRectEncloseOther 114
19.39.1 PROTOTYPE: 115
19.39.2 PARAMETERS: 115
19.39.3 RETURN VALUE: 115
19.39.4 EXAMPLE: 115
19.40 IsRectInsideOther 115
19.40.1 PROTOTYPE: 116
19.40.2 PARAMETERS: 116
19.40.3 RETURN VALUE: 116
19.40.4 EXAMPLE: 116
19.41 IsRectOutsideOther 116
19.41.1 PROTOTYPE: 117
19.41.2 PARAMETERS: 117
19.41.3 RETURN VALUE: 117
19.41.4 EXAMPLE: 117
19.42 IsRectIntersectOther 117
19.42.1 PROTOTYPE: 118
19.42.2 PARAMETERS: 118
19.42.3 RETURN VALUE: 118
19.42.4 EXAMPLE: 118
19.43 IsRectRightUpperToOther 119
19.43.1 PROTOTYPE: 119
19.43.2 PARAMETERS: 119
19.43.3 RETURN VALUE: 119
19.43.4 EXAMPLE: 119
19.44 IsRectRightCenterToOther 120
19.44.1 PROTOTYPE: 120
19.44.2 PARAMETERS: 120

19.44.3 RETURN VALUE: 120
19.44.4 EXAMPLE: 120
19.45 IsRectRightLowerToOther 121
19.45.1 PROTOTYPE: 121
19.45.2 PARAMETERS: 121
19.45.3 RETURN VALUE: 121
19.45.4 EXAMPLE: 121
19.46 IsRectCenterUpperToOther 122
19.46.1 PROTOTYPE: 122
19.46.2 PARAMETERS: 122
19.46.3 RETURN VALUE: 122
19.46.4 EXAMPLE: 122
19.47 IsRectCenterLowerToOther 123
19.47.1 PROTOTYPE: 123
19.47.2 PARAMETERS: 123
19.47.3 RETURN VALUE: 123
19.47.4 EXAMPLE: 123
19.48 IsRectLeftUpperToOther 124
19.48.1 PROTOTYPE: 124
19.48.2 PARAMETERS: 124
19.48.3 RETURN VALUE: 124
19.48.4 EXAMPLE: 124
19.49 IsRectLeftCenterToOther 125
19.49.1 PROTOTYPE: 125
19.49.2 PARAMETERS: 125
19.49.3 RETURN VALUE: 125
19.49.4 EXAMPLE: 125
19.50 IsRectLeftLowerToOther 126
19.50.1 PROTOTYPE: 126
19.50.2 PARAMETERS: 126
19.50.3 RETURN VALUE: 126
19.50.4 EXAMPLE: 126
20. OlvFuzzyRegExpCom 127
20.1.1 DERIVED FROM: 127
20.1.2 CREATION: 127
21. PROPERTIES AND METHODS 127
21.1.1 PROPERTIES READ/WRITE: 127
21.1.2 PROPERTIES READ ONLY: 127
21.2 MatchText : String 127
21.3 MatchBegin : String 127
21.4 MatchEnd : String 127
21.5 MatchDistortion : integer 128
21.6 MatchBeginOffset : integer 128
21.7 MatchEndOffset : integer 128
21.7.1 EXAMPLE: 128
21.7.2 METHODS: 128
21.8 Match 128
21.8.1 DESCRIPTION: 128
21.8.2 PROTOTYPE: 128
21.8.3 PARAMETERS: 128
21.8.4 RETURN VALUE: 129
21.8.5 EXAMPLE: 129
21.9 Grep 129
21.9.1 PROTOTYPE: 129

21.9.2 PARAMETERS: 129
21.9.3 RETURN VALUE: 129
21.9.4 EXAMPLE: 130
21.9.5 NOTE: 130
21.9.6 USAGE EXAMPLE: 130

22. OlvProgressIndicator 130

22.1.1 DESCRIPTION: 130
22.1.2 DERIVED FROM: 131
22.1.3 CREATION: 131
22.1.4 COMMENT: 131

23. PROPERTIES AND METHODS 131

23.1.1 PROPERTIES READ/WRITE: 131
23.2 PageNumber 131
23.2.1 EXAMPLE: 131
23.2.2 PROPERTIES READ ONLY: 131
23.3 TotalPages 131
23.3.1 EXAMPLE: 131
23.3.2 METHODS: 131
23.4 BeginStage 131
23.4.1 DESCRIPTION: 131
23.4.2 PROTOTYPE: 131
23.4.3 PARAMETERS: 132
23.4.4 RETURN VALUE: 132
23.4.5 EXAMPLE: 132
23.5 EndStage 132
23.5.1 PROTOTYPE: 132
23.5.2 PARAMETERS: 132
23.5.3 RETURN VALUE: 132
23.5.4 EXAMPLE: 132
23.6 StageProgress 132
23.6.1 DESCRIPTION: 133
23.6.2 PROTOTYPE: 133
23.6.3 PARAMETERS: 133
23.6.4 RETURN VALUE: 133
23.6.5 EXAMPLE: 133
23.7 StageStatus 134
23.7.1 DESCRIPTION: 134
23.7.2 PROTOTYPE: 134
23.7.3 PARAMETERS: 134
23.7.4 RETURN VALUE: 134
23.7.5 EXAMPLE: 134
23.8 IsActive 134
23.8.1 DESCRIPTION: 134
23.8.2 PROTOTYPE: 134
23.8.3 PARAMETERS: 135
23.8.4 RETURN VALUE: 135
23.8.5 EXAMPLE: 135
23.9 IsCancelled 135
23.9.1 DESCRIPTION: 135
23.9.2 PROTOTYPE: 135
23.9.3 PARAMETERS: 135
23.9.4 RETURN VALUE: 135
23.9.5 EXAMPLE: 135
23.10 ReportLogText, ReportWarning and ReportError 135
23.10.1 DESCRIPTION: 135

23.10.2 PROTOTYPE: 136
23.10.3 PARAMETERS: 136
23.10.4 RETURN VALUE: 136
23.10.5 EXAMPLE: 136

24. OlvRangeCom 136

24.1.1 DERIVED FROM: 136
24.1.2 CREATION: 136

25. PROPERTIES AND METHODS 136

25.1.1 PROPERTIES READ/WRITE: 136
25.2 from:FLOAT 136
25.3 to:FLOAT 136
25.3.1 PROPERTIES READ ONLY: 136
25.3.2 METHODS: 136
25.4 IsInRange: Boolean 137
25.5 IsInFRange : Boolean 137
25.6 IsInTRange : Boolean 137
25.7 IsInFTRange : Boolean 137
25.7.1 PROTOTYPE: 137
25.7.2 PARAMETERS: 137
25.7.3 RETURN VALUE: 137
25.7.4 USAGE EXAMPLE: 137

26. OlvRectCom 138

26.1.1 DESCRIPTION: 138
26.1.2 DERIVED FROM: 138
26.1.3 CREATION: 138

27. PROPERTIES AND METHODS 138

27.1.1 PROPERTIES READ/WRITE: 138
27.2 left 138
27.3 right 138
27.4 bottom 139
27.5 top 139
27.5.1 EXAMPLE: 139
27.5.2 PROPERTIES READ ONLY: 139
27.5.3 METHODS: 139
27.6 Width 139
27.6.1 DESCRIPTION: 139
27.6.2 PROTOTYPE: 139
27.6.3 PARAMETERS: 139
27.6.4 RETURN VALUE: 140
27.6.5 EXAMPLE: 140
27.7 Height 140
27.7.1 DESCRIPTION: 140
27.7.2 PROTOTYPE: 140
27.7.3 PARAMETERS: 140
27.7.4 RETURN VALUE: 140
27.7.5 EXAMPLE: 140
27.8 Square 140
27.8.1 DESCRIPTION: 140
27.8.2 PROTOTYPE: 140
27.8.3 PARAMETERS: 140
27.8.4 RETURN VALUE: 140
27.8.5 EXAMPLE: 141

27.9 IsPointInside 141
27.9.1 DESCRIPTION: 141
27.9.2 PROTOTYPE: 141
27.9.3 PARAMETERS: 141
27.9.4 RETURN VALUE: 141
27.9.5 EXAMPLE: 141
27.10 IsRectInside 142
27.10.1 DESCRIPTION: 142
27.10.2 PROTOTYPE: 142
27.10.3 PARAMETERS: 142
27.10.4 RETURN VALUE: 142
27.10.5 EXAMPLE: 142
27.11 IsIntersect 143
27.11.1 DESCRIPTION: 143
27.11.2 PROTOTYPE: 143
27.11.3 PARAMETERS: 143
27.11.4 RETURN VALUE: 143
27.11.5 EXAMPLE: 143
27.12 IntersectRect 143
27.12.1 DESCRIPTION: 143
27.12.2 PROTOTYPE: 143
27.12.3 PARAMETERS: 143
27.12.4 RETURN VALUE: 143
27.12.5 EXAMPLE: 143
27.13 Intersect2Rects 144
27.13.1 DESCRIPTION: 144
27.13.2 PROTOTYPE: 144
27.13.3 PARAMETERS: 144
27.13.4 RETURN VALUE: 144
27.13.5 EXAMPLE: 144
27.14 UniteRect 144
27.14.1 DESCRIPTION: 144
27.14.2 PROTOTYPE: 144
27.14.3 PARAMETERS: 144
27.14.4 RETURN VALUE: 144
27.14.5 EXAMPLE: 145
27.15 UnitePoint 146
27.15.1 DESCRIPTION: 146
27.15.2 PROTOTYPE: 146
27.15.3 PARAMETERS: 146
27.15.4 RETURN VALUE: 146
27.15.5 EXAMPLE: 146
27.16 OffsetRect 147
27.16.1 DESCRIPTION: 147
27.16.2 PROTOTYPE: 147
27.16.3 PARAMETERS: 147
27.16.4 RETURN VALUE: 147
27.16.5 EXAMPLE: 147
27.17 InflateRect 148
27.17.1 DESCRIPTION: 148
27.17.2 PROTOTYPE: 148
27.17.3 PARAMETERS: 148
27.17.4 RETURN VALUE: 148
27.17.5 EXAMPLE: 148
27.18 IsRectSimilarTo 148
27.18.1 DESCRIPTION: 148
27.18.2 PROTOTYPE: 148

27.18.3 PARAMETERS: 148
27.18.4 RETURN VALUE: 149
27.18.5 EXAMPLE: 149
27.19 IsEqual 149
27.19.1 PROTOTYPE: 149
27.19.2 PARAMETERS: 149
27.19.3 RETURN VALUE: 149
27.19.4 EXAMPLE: 149
28. OlvTextStyleInfo 150
28.1.1 DESCRIPTION: 150
28.1.2 DERIVED FROM: 150
28.1.3 CREATION: 150
28.2 PROPERTIES AND METHODS 150
28.2.1 PROPERTIES READ/WRITE: 150
28.2.2 PROPERTIES READ ONLY: 150
28.3 FontName : string 150
28.4 FontSize : float 150
28.5 FontSizeDeviation : float 151
28.6 FontFlags : integer 151
28.7 NumOfPages : integer 151
28.8 NumOfTextBlocks : integer 151
28.9 NumOfTextLines : integer 151
28.10 NumOfQuads : integer 151
28.10.1 EXAMPLE: 151
28.10.2 METHODS: 151
29. OlvLinkCom 152
29.1.1 DERIVED FROM: 152
29.1.2 CREATION: 152
29.2 PROPERTIES AND METHODS 152
29.2.1 PROPERTIES READ/WRITE: 152
29.3 LinkType: number 152
29.4 TargetPageNumber:number 152
29.5 TargetString:string 152
29.6 TargetBox:OlvFRectCom 152
29.7 TargetZoomBox:OlvFRectCom 153
29.8 ZoomToPageWidth:boolean 153
29.9 Properties Read only: 153
29.9.1 METHODS: 153
29.9.2 USAGE EXAMPLES: 153
30. Appendix A: Script Files 153
30.1 XMDTeplate.js 153
30.2 XMD4Include.js 155

1. Olive Recognition Language (ORL) SDK

Olive ViewPoint is a system that transforms scanned and legacy electronic documents to an advanced, platform independent XML electronic format.

ViewPoint performs the transformation process in stages:

1.1 Preprocessing

ViewPoint prepares the document for segmentation by straightening and cleaning scanned documents, performing OCR, and extracting metadata.

1.2 Segmentation

The segmentation process identifies document components, such as headers and footers, headlines, body text, paragraphs, figures, and captions. Each component is tagged and saved with its textual content in an XML file of the document page and the entire document.

1.3 Warehouse

The ViewPoint Warehouse is a document repository that can be centrally accessed from remote locations. The Warehouse distributed over a network and accessed from any computer on the network.

1.4 Publication

ViewPoint documents can be accessed directly, displayed in Web applications, or repurposed into databases and other applications.

(continued in XMD Template)

2. How to Write ORL Scripts

Olive ViewPoint includes a toolkit of scripts and objects that enable JavaScript programmers to rapidly and accurately write ORL scripts, which includes a the following tools:

- **XMDScripts folder** contains scripts used by other scripts.
- **Samples folder** contains sample scripts.
- **AP subfolder** contains generic workflow scripts.
- **XMDTemplate.js** is a JavaScript file that provides a framework to customizing with the required functions. It functions as the "umbrella" script that calls the other sub-scripts and objects.

- **XMD4Include.js** is a JavaScript file that includes many of the most commonly used ORL functions. The XMDTemplate script may call the XMD4Include script to perform certain common functions, or it may call other sub-scripts or ORL objects as required.
- **TextStatistics.js** is a JavaScript that implements the TextStyleProcessor object that is used for style analyses. Its standard behavior can be customized to enable:
    - Changing **Members** of the **Params** structure
    - Substitute any function of the **TextStyleProcessor** object
    - **TextStyleProcessor** object has three empty functions for custom additions
- **ORL Objects** are C++ objects that are programmed to perform the specific ORL functions.

The following sections follow the workflow to write ORL scripts.

2.1 XMDTemplate.js

The JavaScript file **XMDTemplate.js** is ViewPoint ORL's template script that serves as the framework from which custom scripts are written. It can include specific functional code, or functioning to operate other sub-scripts and ORL objects.

This script is built of a generic "*framework*", which includes all the necessary global definitions defined in XMD4Include.js, error handling, logging and progress indicator.

Its main feature is the **MainCode()** placeholder section. All custom code for specific functions and calls to operate other scripts or objects are written into the **MainCode()** core.

The critical part of the XMDTemplate.js file is here:

```
…
{
   // 'ScriptProcess' can be replaced by any name for a stage
   eval( GetGlobalDefinitions() );
   StartScript();

   MainCode(); // ... To be filled with user's desired code ...

   EndScript();
}
catch( exception )
{
   if (exception.description == "OlvException: Failed to load include
file")
         OnIncludeFileException();
   else
         OnException( exception ); // Exception handling
}
```

```
function OnIncludeFileException(description)
{
   OlvProgressIndicator.BeginStage(strStage, 0, true);
   OlvProgressIndicator.ReportError(strGlobalMsg);
   OlvProgressIndicator.EndStage(strStage);
}

//================================================================
// To be filled with user's desired code
//================================================================
function MainCode()
{
   // ... TBD ...
}
...
```

The highlighted sections above indicate where to enter the specific functional code for the script.

See Appendix A for the full content of the **XMDTemplate.js** file.

2.2 XMD4Include.js

The other JavaScript file is **XMD4Include.js**, which contains global definitions (including reserved names) and Flags for *OlvPropertySet* records and some very useful functions. It functions along side **XMDTeplate.js** and the **ORL objects**.

2.2.1 Reserved Names of OlvPropertySet records

Reserved names are set and/or processed in built-in processes, such as *Basic Segmentation*, *TOC Recognition*, *Header/Footer Recognition*, *Picture Recognition* and *Final Segmentation*, and by ORL scripts.

Listed below are reserved names and properties, with a short explanation of each:

OlvPropName_FILE_PATH

- Value is a string.
- Is set by Basic Segmentation process for OlvDocObject.
- Present full path to source document

OlvPropName_PRIMITIVE_IGNORE

- Value is TRUE or FALSE. If TRUE then Final Segmentation does not create an XMDPrimitive.

OlvPropName_PRIMITIVE_ID

- Value is string.
- OlvBlocks with different values of "PRIMITIVE_ID" are not supposed to be united to the same XMDPrimitive by Final Segmentation.

OlvPropName_PRIMTIVE_TAG_ID

- Value is string. It should be one of PrimitiveTagIDs defined in XMD.ini or Profile in [PrimitiveTag]
- For example: "primtAgency", "primtBodyTitle", "primtAuthorName" - one
- The OlvTag should be used by Final Segmentation to create an XMDPrimitiveTag with the Name = <value of property>

OlvPropName_TAG_EXTRACT_AS_PRIMITIVE

- Value does not matter.
- Presence of this property means that OlvTag should be extracted by Final Segmentation to a separate XMDPrimitive

OlvPropName_TAG_LINK_PAGE_NO

- Value is integer.
- The property should be set for OlvTagCom object only.
- Final Segmentation creates Link to Page with Source Box = Box(OlvTag) and PageNumber = <value of property>

OlvPropName_TAG_LINK_PAGE_LABEL

- Value is string
- The property should be set for OlvTagCom object only.
- Final Segmentation creates Link to Page with Source Box = Box of OlvTag and PageLabel = <value of property>

OlvPropName_TAG_LINK_PAGE_TARGET_BOX

- Value is rectangle.
- The property should be set for OlvTagCom object only.
- One of two following properties should be must be present: OlvPropName_TAG_LINK_PAGE_LABEL or OlvPropName_TAG_LINK_PAGE_NO
- Final Segmentation creates "See Box" Link with Target Box = <value of Property>

OlvPropName_PAGE_FOOTER

- Value does not matter.
- Presence of this property means that OlvBlock is part of Page Footer.
- The property is set by Footer/Header recognition process

OlvPropName_PAGE_HEADER

- Value does not matter.
- Presence of this property means that OlvBlock is part of Page Header.
- The property is set by Footer/Header recognition process

OlvPropName_PAGE_TOC

- Value does not matter.
- Presence of this property means that Page contains a TOC area.
- The property is set by TOC recognition process

OlvPropName_TOC_AREA

- Value does not matter.
- The property should be set for OlvBlockCom object only.
- Presence of this property means that Block is part of TOC Area.

OlvPropName_TOC_HEADING_LEVEL

- Value is Integer.
- Represent level of TOC item
- The property is set by TOC recognition process to the OlvTextBlock or OlvTag object created from text matching a TOC item

OlvPropName_PAGE_LABEL

- Value is string
- The property should be set for OlvPageCom object only.
- FinalSegmentation sets Page Label = <value of Property>

OlvPropName_PAGE_SECTION

- Value is string.
- The property should be set for OlvPageCom object only.
- FinalSegmentation sets Page Section = <value of Property>

OlvPropName_ENTITY_ID_ALL

- Value is a string.
- All and only OlvBlocks having this property with the same value and only they are included to the same Entity by Final Segmentation.
- Different chunks (continuations) are to be created for case of different pages.

OlvPropName_ENTITY_ID_PART

- Value is a string.
- All OlvBlocks having this property with the same value are included to the same Entity by Final Segmentation.
- Different chunks (continuations) are created for case of different pages.

OlvPropName_PARENT_ENTITY_ID

- Value is a string.

- Defines target XMDEntity to be embedded by Final Segmentation to another one with an equal ENTITY_ID value.

PARENT_PRIMITIVE_ID

- Value is a string.
- It defines that target XMDEntity should be embedded by Final Segmentation to a primitive with an equal PRIMITIVE_ID value.

OlvPropName_PRIMITIVE_SEQ_NO

- Value is integer
- All XMDPrimitives of the same XMDEntity are sorted by Final Segmentation according this value.

OlvPropName_LINK_FROM_PDANNOT

- Value is Boolean
- Basic Segmentation sets this property to OlvLink an object if a PDF object created it. If there are two OlvLink objects, then Final Segmentation prefers to create an XMDLink object from one that has this property.

OlvPropName_DRAWING

- Value is integer
- Basic Segmentation sets this property to an OlvGraphBlock object if it is created from vector graph objects of a PDF document.

OlvPropName_PHOTO

- Value is integer
- Basic Segmentation sets this property to an OlvGraphBlock object if it is created from image objects of a PDF document.

OlvPropName_PICTURE_ID

- Value is string
- This property is set to Picture and to all related OlvTextBlock objects (caption and text) by the Picture Recognition process.
- ORL Scripts use this property to filter Picture related text.

2.2.2 Reserved Flags of OlvPropertySet records

The following flags are processed by Final Segmentation.

- **OlvFlag_Temporary** - The property is temporary and is not saved in PDF
- **OlvFlag_Attribute** - Value of Property is 'confidence' (float from [0.1])
- **OlvFlag_Display** – Acrobat View displays the Name and Value of this Property in the OlvBlock Label.

- **OlvFlag_PredefinedArea** – “PropName” is a type that must be listed in XMD.ini or Profile in [PreDefAreas] : AreasTypes. Value of OlvProperty is the box that defines the page’s predefined area. Basic segmentation adds to the OlvPage object one property for each predefined area.
- **OlvFlag_Internal** – used for internal XMD purposes

Presence of MD Flag means that the string value of the property is metadata for the relevant level

- **OlvFlag_MD4Doc** – can be present in OlvPropertySet of OlvDocCom
- **OlvFlag_MD4Page** – can be present in OlvPropertySet of OlvPageCom
- **OlvFlag_MD4Entity** – can be present in OlvPropertySet of OlvBlockCom
- **OlvFlag_MD4Primitive** – can be present in OlvPropertySet of OlvBlockCom
- **OlvFlag_MD4Tag** – can be present in OlvPropertySet of OlvTagCom

The following flags can be present only in Property of TextBlock.

- **OlvFlag_EntType** – PropName must be one of ENTITY_TAG_NAMEs: “article”, “picture” or “ad”.
- **OlvFlag_EntSubType** – PropName must be one of SUB_ENTITY_TAG_NAMEs: “Table” etc.
- **OlvFlag_ElemType** – PropName must be one of ELEMENT_TAG_NAMEs: “article.content”, ”article.hedLine.hl1” etc.
- **OlvFlag_ElemSubType** – PropName must be one of SUB_ELEMENT_TAG_NAME: “Chapter”, “Paragraph” etc. Note that together with this flag the TextBlock must contain property with OlvFlag_ElemType otherwise the flag is inored by Final Segmentation

2.3 Predefined values and functions to be used in in XMD4Include.js scripts

2.3.1 Enumerations used by methods as return values or arguments

Source type is type of OlvDoc.SourceType property

- var OlvSourceType_UNKNOWN = "Undefined";
- var OlvSourceType_SCANNED = "Scanned";
- var OlvSourceType_DIGITAL = "Digital";
- var OlvSourceType_MIXED = "Mixed";
- var OlvSourceType_INVALID= "Invalid";

Block type is returned by method OlvBlock.GetTypeNumeric().

BlockType amd OR combinations of several BlockTypes is a OlvBlockFilterCom.Type property's type.

- var OlvBlockTypeUnknown = 0;
- var OlvBlockTypeText = 1;
- var OlvBlockTypeGraph = 2;
- var OlvBlockTypeVerSep = 4;
- var OlvBlockTypeHorSep = 8;
- var OlvBlockTypeTable = 16; //0x10
- var OlvBlockTypeAll = 255; //0xff

Block Type S is a type returned by method OlvBlock.GetType().

- var OlvBlockTypeUnknownS = "UnknownBlock";
- var OlvBlockTypeTextS = "TextBlock";
- var OlvBlockTypeGraphS = "GraphBlock";
- var OlvBlockTypeVerSepS = "VerSeparator";
- var OlvBlockTypeHorSepS = "HorSeparator";
- var OlvBlockTypeTableS = "TableBlock";

Orientation is an argument's type of method OlvTextBlockCom.InitEx() method.

- var OlvOrientation_Unknown = 0;
- var OlvOrientation_Portrait = 1;
- var OlvOrientation_Landscape = 2;

Order is a OlvBlockFilterCom.OrderRegime property's type and an argument's type of

OlvBlockSortingCom.SortBlocks() method.

- var OlvUndefinedOrder = 0;
- var OlvReadingOrder = 1; // Sort by reading order of blocks
- var OlvPageNumTopOrder = 2; // Sort by Page Number and then by Top Order

Page Status is a OlvBlockFilterCom.PageStatus property's type

- var OlvUndefinedPageStatus = 0;

- var OlvPageSingle = 1;
- var OlvPageLeft = 2;
- var OlvPageRight = 3;

Order Relation is a OlvBlockFilterCom.OrderRelations property’s type

- var OlvOrdUndefined = 0;
- var OlvOrdBefore = 1;
- var OlvOrdOverlap = 2;
- var OlvOrdInside = 3;
- var OlvOrdAfter = 4;

Location Relation and any OR combination of several Location Relations is a OlvBlockFilterCom.LocationRelations property’s type

- var OlvLocUndefined = 0x0000;
- var OlvLocAbove = 0x0001;
- var OlvLocBelow = 0x0002;
- var OlvLocLeft = 0x0004;
- var OlvLocRight = 0x0008;
- var OlvLocHorCenter = 0x0010;
- var OlvLocVerCenter = 0x0020;
- var OlvLocInside = 0x0040;
- var OlvLocOverlap = 0x0080;
- var OlvLocOutside = (OlvLocAbove | OlvLocBelow | OlvLocLeft | OlvLocRight); // Above or Below or Left or Right

Default name to be displayed in Progress Indicator Caption while running script.

- var strStageNameGlobal = "Executing Script";

Default section in Profile and DSC containing parameters for custom scripts

- var sScriptParametersSectionName = "SCRIPT_PARAMETERS";

2.3.2 Functions toArray() and toSafeArray()

Collections returned by ORL functions or passed as parameters are SafeArray objects.

Java Script collections have structure of Scripting.Dictionary object

XMD4Include.js contains two functions for conversion between these objects.

- function toArray(safeArray)

Sample:

- var arrAllTBlocksInPage = toArray(Page.GetAllTextBlocks());
- function toSafeArray(array)

Sample:

- var arrCandidates = toSafeArray(arrAllTBlocksInPage);
- var arrSortedCandidates = toArray(sorter.SortBlocks(arrCandidates, OlvReadingOrder));

2.4 Standard scripts of the generic workflow

2.4.1 AP_FindHeadlines.js

- Looks for document headlines
- Looks for the first group of Text Blocks that look like document's title.
- Set the property "article.headLine.hl1" with flags OlvFlag_ElemType and OlvFlag_Attribute and set to Document a property "dc:title" with the flag OlvFlag_MD4Doc, whose value is the contents of these blocks.

2.4.2 AP_RevisePictureRelatedBlocks.js

- This script performs Picture Recognition by trying to find picture related text as captions.
- It expands Image OlvBlocks by searching for nearby text, looks for Picture Captions and Picture Text blocks and sets the ENTITY_ID_ALL, PICTURE_ID, "picture", "picture.caption" and "picture.text" properties to the relevant OlvBlocks

2.4.3 AP_HierarchyRecognition.js

- Does nothing if hierarchy has been previously recognized in the document, when the function IsHierarchyPresent() returns TRUE.
- It uses the TextStyleProcessor object described in the TextStatistics.js to find level 1 and 2 headings and marks them with OlvPropName_TOC_HEADING_LEVEL properties.
- It does not assign element types to headings.

2.4.4 AP_AssignElementTypesToHeadings.js

This script assigns text blocks with element types (and subtypes) according the TOC_HEADING_LEVEL property (0=Title, 1=chapter, etc...)

2.4.5 AP_SetPageBasedEntities.js

- Does nothing if hierarchy has been previously recognized in the document, when the function IsHierarchyPresent() returns TRUE.
- Creates page-based hierarchy – each page is a separate entity (page-based segmentation).
- This script sets the property OlvPropName_ENTITY_ID_PART with the value=<Page_No> to all OlvTextBlock on each page. If the parameter [SCRIPT_PARAMETERS] : “bIC_IgnoreEmbeddedBlocks” is 1 then embedded blocks are ignored.
- It also sets the property OlvPropName_DOC_HIERARCHY_IS_PAGE_BASED to the OlvDoc object.

2.4.6 AP_SetEntitiesFromTOCHeadings.js

- Separates document to entities - makes a new entity starting from each heading (chapter, paragraph, etc.).
- It classifies OlvTextBlocks into separate groups, to unite them into separate Entities in Final Segmentation. A unique ID for each Entity is generated and set a value for the property OlvPropName_ENTITY_ID_PART to all OlvTextBlocks belonging to that Entity. Classification is based on the property BSPropName_TOC_HEADING_LEVEL.
- Parameters from section [SCRIPT_PARAMETERS] define classification criteria.

2.5 ORL Objects Overview

Several types of objects may be used in ORL scripts:

- General
- Basic
- Document-Oriented
- Functional

2.5.1 General Objects

OlvFRectCom

Rectangle object is identifies a rectangle by the top left and bottom right corners.

OlvFPointCom

Float Point object defines the X and Y coordinates of each corner of a rectangle.

OlvRangeCom

Range is the interval between two numbers, such as the distance (number of pixels) between rectangle borders (the rectangle's width or height).

OlvFuzzyCompareCom

Fuzzy or approximate **comparison** of rectangles, points, ranges and numbers

OlvFuzzyRegExpCom

Fuzzy or approximate search of **regular expression** in text

2.6 Basic Objects

OlvPropertySet

- Record set of the following structure: {PropertyName, PropertyValue, Flags}
- No instance of this object can be created.
- All Document-oriented objects are derived from it.

ElementProvider

OlvBlockCom

2.6.1 Document-Oriented Objects

OlvDocCom

OlvTextStyleInfo

OlvPageCom

OlvTextStyleGallery

OlvTextBlockCom

OlvGraphBlockCom

OlvVSeparatorBlockCom

OlvHSeparatorBlockCom

OlvTableBlockCom

OlvTagCom

OlvTextLineCom

OlvQuadCom

2.6.2 Functional Objects

OlvBlockFilterCom

OlvBlockSortingCom

OlvProgressIndicatorCom

3. ORL Hierarchy

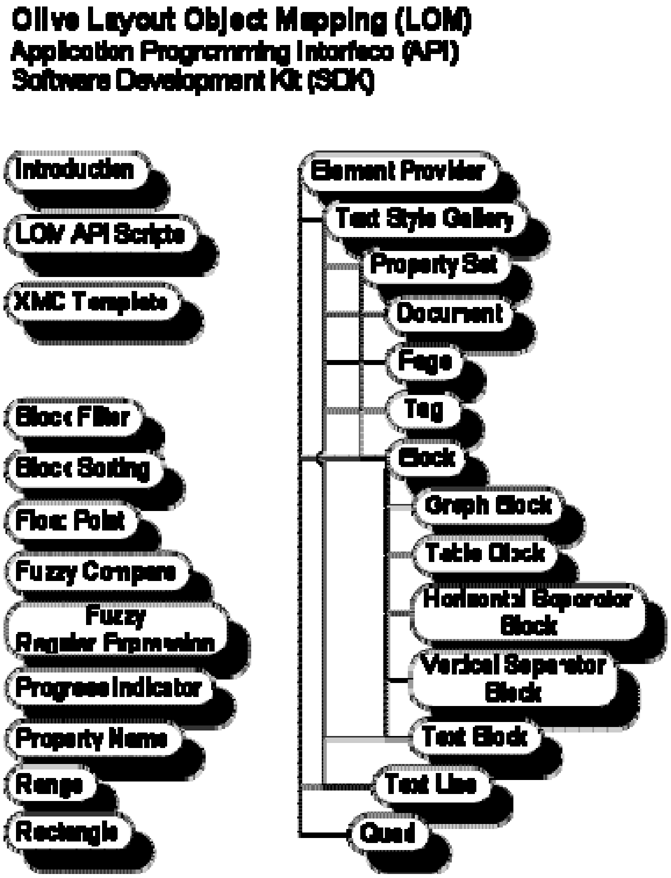
Some of the modules in the ORL API are hierarchal, meaning that some modules are nested, or they reference other, higher level modules.

This hierarchy is based on the structure of documents, where a complete document may contain several pages. Each page, in turn, contains individual

elements, such as text boxes, pictures, or separator lines, and each of these elements may have different characteristics, such as location on the page, tag, text style, and hierarchy in document and within entity.

The ORL API modules that relate to elements are nested under the Element Provider module, as shown.

Continue to Element Provider



3.1 Sample Scripts

4. OlvTextStyleGallery

4.1.1 Description:

The Text Style Gallery is a class that is intended to be inherited by text containing classes, such as: OlvDocCom, OlvPageCom, OlvTextBlockCom, and OlvTextLine. It is a collection of text-styles (OlvTextStyleInfo) and some statistics, all referring to the specific (current) text of the object inheriting it.

4.1.2 Derived from:

4.1.3 Creation:

An **abstract** interface - e.g. an object of this class can't be created as is, but it can be part of an inheritor object.

4.2 PROPERTIES AND METHODS

4.2.1 Properties Read/Write:

4.2.2 Properties Read only:

4.3 NumberOfTextStyles : integer

Number of text-styles (OlvTextStyleInfo) in the specific (current) text.

4.4 NumberOfCharacters : integer

Number of characters (length) of the specific (current) text.

4.5 NumberOfLetters : integer

Number of alpha-bet characters in the specific (current) text.

4.6 NumberOfCapitals : integer

Number of capital alpha-bet characters in the specific (current) text.

4.7 NumberOfDigits : integer

Number of digit characters in the specific (current) text.

4.7.1 Example:

```
var num = Doc.NumberOfTextStyles; // Total text-styles in the
OlvDocCom

num = Page.NumberOfCharacters; // Total characters in the OlvPageCom

num = TextBlock.NumberOfCapitals; // Total capitals in the
OlvTextBlockCom

num = TextLine.NumberOfLetters; // Total alpha-bet characters in the
OlvTextLine

num = Doc.NumberOfDigits; // Total digits in the OlvDocCom
```

4.7.2 Methods:

4.7.3 GetTextStyleInfo

4.7.4 Description:

Retrieves the OlvTextStyleInfo number “lIndex” (see Parameters) .

4.7.5 Prototype:

```
GetTextStyleInfo( lIndex : integer ) : OlvTextStyleInfo.
```

4.7.6 Parameters:

lIndex : integer – There is no special meaning to that index. It is the internal index of the OlvTextStyleInfo to be retrieved out of **NumberOfTextStyles** possible text-styles of the specific (current) text. It can be used to loop trough all the text-styles of the specific (current) text.

4.7.7 Return value:

See description.

4.7.8 Example:

```
//=======================================

// This code checks if current page has big fonts

//=======================================

var bPageHasBigFonts = false;
```

```
// Get number of text-styles in the current page (OlvPageCom)

var TotalTextStyles = Page.NumberOfTextStyles;

// Loop through all the text-styles of current page

for (var i = 0; i < TotalTextStyles; ++i)

{

// get current text-style

var TextStyleInfo = Page. GetTextStyleInfo(i);

// Is it a big font?

   if( TextStyleInfo.FontSize > 16.0 )

bPageHasBigFonts = true;

}
```

4.7.9 GetPopularTextStyleInfo

4.7.10 Description:

Retrieves the OlvTextStyleInfo of the most common (popular) Text-Style of the specific (current) text. A Text-Style's popularity, in a specific (current) text, is the number quads (OlvQuadCom) in that text, written in that style.

4.7.11 Prototype:

```
GetPopularTextStyleInfo() : OlvTextStyleInfo.
```

4.7.12 Parameters:

4.7.13 Return value:

See description.

4.7.14 Example:

```
// Find the name of the most common (Popular) text-style in current
page var PopularTextStyleInfo = Page. GetPopularTextStyleInfo();  var
CommonFontName = PopularTextStyleInfo.FontName;
```

5. OlvPropertySet

OlvPropertySet is an "abstract" object, providing an interface to the Properties of the following objects: OlvDocCom, OlvPageCom, OlvBlockCom, OlvTagCom, OlvTextLineCom and OlvQuadCom.

OlvPropertySet is a set of properties with the following structure:

{**Name**, **Value**, **Flags**}.

Name is a string and is used as a key. OlvPropertySet can contain several properties having the same name. See **list of reserved Property Names** in script **XMD4Include.js**.

Value can be of any basic type (number, string, or boolean) or a custom object:

OlvFRect, OlvFPoint and OlvFRange.

Flags are "OR" combinations of predefined flags.

See **list of reserved Property Flags** in script **XMD4Include.js**.

Script can use OlvPropertySet to keep information in processed objects. Note that Properties having unreserved names are ignored outside of the script.

5.1.1 Derived from:

5.1.2 Creation:

5.2 PROPERTIES AND METHODS

5.2.1 Properties Read/Write:

5.2.2 Properties Read only:

5.2.3 Methods:

5.3 SetProperty

Adds a property with a specified name, value and flags to source object's OlvPropertySet. If the object had one or more properties with the same name, they are deleted.

5.3.1 Prototype:

```
SetProperty propName : String, propValue : Object, propFlags :
Number)
```

5.3.2 Parameters:

propName: A string of the property's name.

propValue: An object that represents any basic type value (number, string, boolean) or a custom object: OlvFRect, OlvFPoint and OlvFRange

propFlags: An integer number that is "OR" combination of flags.

5.3.3 Return value:

5.3.4 Example:

```
OlvDoc.SetProperty("VERSION", "3.1", OlvFlag_MD4Doc);
```

5.4 RemoveProperty

Removes all Properties with a specified name from source object's OlvPropertySet

5.4.1 Prototype:

```
RemoveProperty (propName : String) : Boolean
```

5.4.2 Parameters:

propName - A string that means name of property.

5.4.3 Return value:

TRUE if the property was found and successfully removed; otherwise **FALSE**.

5.4.4 Example:

```
Var bIsRemoved = OlvDoc.RemoveProperty("VERSION");
```

5.5 ClearAllProperties

Removes all properties with at least one specified flag from thesource object's OlvPropertySet.

5.5.1 Prototype:

```
ClearAllProperties ( propFlags : Number)
```

5.5.2 Parameters:

propFlags - Integer number that is “OR” combination of flags.

To remove all properties you should use propFlags= -1 (0xffffffff)

5.5.3 Return value:

5.5.4 Example:

See CountProperties.

5.6 CountProperties

Counts the number of properties in source object’s OlvPropertySet

5.6.1 Prototype:

```
CountProperties() : Number
```

5.6.2 Parameters:

5.6.3 Return value:

Number of Properties

5.6.4 Example:

```
Var nProps = OlvDoc.CountProperties(); // n = 0
OlvDoc. SetProperty("VERSION”, “3.1”, OlvFlag_MD4Doc);
OlvDoc. SetProperty("HOLD”, true, OlvFlag_Temporary);
Var bIsTemp = OlvDoc. IsPropertyTemporary(“HOLD”); // True
OlvDoc. SetProperty("HOLD_BY”, “Mike”, OlvFlag_Temporary);
nProps = OlvDoc.CountProperties(); // n = 3
OlvDoc. ClearAllProperties(OlvFlag_Temporary);
nProps = OlvDoc.CountProperties(); // n = 1
var bExists = OlvDoc. IsPropertyExists("HOLD_BY”); // False
```

5.7 IsPropertyTemporary

Checks if the first property with the specified name contains the flag OlvFlag_Temporary.

5.7.1 Prototype:

```
IsPropertyTemporary (propName : String) : Boolean
```

5.7.2 Parameters:

propName: A string that means name of property.

5.7.3 Return value:

TRUE if at least one property with the specified name is present and the first one contains OlvFlag_Temporary;

Otherwise **FALSE**.

5.7.4 Example:

See CountProperties.

5.8 IsPropertyExists

Checks if a property with the specified name is in the source object's OlvPropertySet.

5.8.1 Prototype:

```
IsPropertyExists (propName : String) : Boolean
```

5.8.2 Parameters:

propName: A string that defines the property's name.

5.8.3 Return value:

TRUE if at least one property with the specified name is found; otherwise **FALSE**.

5.8.4 Example:

See CountProperties.

5.9 GetPropertyFlags

Gets the flag values for the first property found with the specified name.

5.9.1 Prototype:

```
GetPropertyFlags (propName : String) : Number
```

5.9.2 Parameters:

propName - A string that defines the property's name.

5.9.3 Return value:

Numberic value if property exists; otherwise 0.

5.9.4 Example:

```
OlvDoc. SetProperty("VERSION", "3.1", 20);

OlvDoc. SetProperty("VERSION", "4.1", 10);

Var flags = OlvDoc.GetPropertyFlags("VERSION"); // 20
```

5.10 GetPropertyValue

Gets the value of the first found property with the specified name.

5.10.1 Prototype:

```
GetPropertyValue (propName : String) : Object
```

5.10.2 Parameters:

propName - A string that means name of property.

5.10.3 Return value:

Value of the first found property if property exists; otherwise null. Value can be String, Number, Boolean, ColvFRect, ColvFPoint or ColvFRange.

5.10.4 Example:

```
OlvDoc. SetProperty("VERSION", "4.1", 10);

Var flags = OlvDoc.GetPropertyFlags("VERSION"); // = 20

Var flags = OlvDoc.GetPropertyFlags("ZZZ"); // = null
```

5.11 SetMultiProperty

Adds one or more properties with a specified name, values and flags to the source object's OlvPropertySet. If the object has properties with the same, name they are deleted.

5.11.1 Prototype:

```
SetMultiProperty (propName : String, propValueArr : VBArray,
propFlagsArr : VBArray)
```

5.11.2 Parameters:

propName - A string that means name of property.

propValueArr - A VBArray containing values of properties.

propFlagsArr - A VBArray containing values (numbers) of flags of properties.

If its size is smaller than size of propValueArr then absent flags are set to 0. If the size is bigger then extra flags are ignored.

5.11.3 Return value:

5.11.4 Example:

```
Block.ClearAllProperties(-1);

PropNum = Block.CountProperties(); // = 0

// SetMultiProperty value var arrValues = new Array(12.5, 22, 40);

var arrFlags = new Array();

arrFlags.push(OlvFlag_Temporary + OlvFlag_Attribute);

arrFlags.push(OlvFlag_Attribute);

Block.SetMultiProperty("COUNT", toSafeArray(arrValues),
   toSafeArray(arrFlags));

PropNum = Block.CountProperties(); // 3

// Get Values

  var safeArrVal = Block.GetMultiPropertyValues("COUNT");

var arrValues = toArray(safeArrVal);

var a0 = arrValues[0]; // 12.5

var a1 = arrValues[1]; // 22

var a2 = arrValues[2]; // 40

// Get Flags

  var safeArrFlags = Block.GetMultiPropertyFlags("COUNT");

var arrFlags = toArray(safeArrVal);

var a0 = arrFlags[0]; // = 3 OlvFlag_Temporary + OlvFlag_Attribute

var a1 = arrFlags[1]; // = 2 OlvFlag_Attribute
var a2 = arrFlags[2]; // = 0
```

5.12 GetMultiPropertyValues

Get all values of Properties with specified name.

5.12.1 Prototype:

```
GetMultiPropertyValues (propName : String) : propValueArr : VBArray
```

5.12.2 Parameters:

propName - A string that means name of property.

5.12.3 Return value:

propValueArr - A VBArray containing values of properties. The returned array is empty If no property found.

5.12.4 Example:

See SetMultiProperty

5.13 GetMultiPropertyFlags

Get all flags of Properties with specified name.

5.13.1 Prototype:

```
GetMultiPropertyFlags (propName : String) : propFlagsArr : VBArray
```

5.13.2 Parameters:

propName - A string that means name of property.

5.13.3 Return value:

propFlagsArr - A VBArray containing flags of properties. The returned array is empty if no property found.

5.13.4 Example:

See SetMultiProperty

5.14 AddMultiProperty

5.14.1 Description:

Add a Property with specified name, value and flags to source object’s OlvPropertySet. Note that if object had one or more properties with the same name they are not deleted.

5.14.2 Prototype:

```
AddMultiProperty (propName : String, propValue : Object, propFlags :
Number)
```

5.14.3 Parameters:

propName - A string that means name of property.

propValue - An object that represents any basic type value (number, string, boolean) or one of custom objects: OlvFRect, OlvFPoint and OlvFRange

propFlags - Integer number that is “OR” combination of flags.

5.14.4 Return value:

5.14.5 Example:

```
OlvDoc.AddMultiProperty("VERSION”, “3.1”, OlvFlag_MD4Doc);
```

5.15 AddPropertyFlags

This method adds flags to all Properties of source object’s OlvPropertySet with a specified name.

5.15.1 Prototype:

```
AddPropertyFlags (propName : String, propFlags : Number)
```

5.15.2 Parameters:

propName - A string of the property’s name

propFlags - Integer number that is “OR” combination of flags.

5.15.3 Return value:

5.15.4 Example:

```
OlvDoc.AddPropertyFlags("VERSION”, OlvFlag_Display | OlfFlag_MD4Doc);
```

5.16 ErasePropertyFlags

This method removes flags from all Properties of source object's OlvPropertySet with a specified name.

5.16.1 Prototype:

```
ErasePropertyFlags (propName : String, propFlags : Number)
```

5.16.2 Parameters:

propName - String of the property's name

propFlags - Integer number that is "OR" combination of flags.

5.16.3 Return value:

5.16.4 Example:

```
OlvDoc.ErasePropertyFlags("VERSION", OlvFlag_Display);
```

6. OlvDocCom

OlvDocCom is used when working with a complete document, of which only one instance exists per document. The calling program sends it to the script as the global variable **OlvDoc**.

6.1.1 Derived from:

OlvPropertySet

OlvTextStyleGallery

6.1.2 Creation:

6.2 PROPERTIES AND METHODS

6.2.1 Properties Read/Write:

6.2.2 Properties Read only:

6.3 NumberOfPages : Integer

Integer - number of pages in the Document

6.3.1 Example:

```
Var Num = OlvDoc.NumberOfPages;
```

6.4 SourceType : String

Specifies the source document type.

Can be one of following strings:

- OlvSourceType_**UNKNOWN** - document type is unknown
- OlvSourceType_**SCANNED** - all pages are scanned
- OlvSourceType_**DIGITAL** – all pages are digital
- OlvSourceType_**MIXED** – both scanned and digital pages in document
- OlvSourceType_**INVALID** – program error

6.5 HasTOCPages : Boolean

TRUE if at least one of the document's pages contains a "TOC_PAGE" property in its PropertySet.

6.6 IncludePath : String

Path to the XMD4Include.js script that contains predefined values and definitions. XMDTemplate.js script uses this property to function.

It is recommended to write any script using XMDTemplate.js as its template.

Path to XMD4Include.js containing predefined values and definitions. XMDTemplate.js script uses this property to perform including.

6.7 DocPath : String

A string that contains the path of the current document.

6.8 Methods:

6.9 GetExternalParameter

This parameter is read from **XMD.ini** and **profile.ini** files, and its value is returned as a string. In batch processing, the **DSC** file is also read.

6.9.1 Prototype:

```
GetExternalParameter( sSectionName : String, sParameterName : String
) : String
```

6.9.2 Parameters:

sSectionNam, **sParameterName -** Strings, the names of section and parameter as written in **XMD.ini** and **profile.ini** files, or in the **DSC** file of the Job Manager.

6.9.3 Return value:

String containing the parameter's value as been written in XMD.ini and profile.ini. (default = "").

6.9.4 Example:

```
sParameterContent = OlvDoc.GetExternalParameter( "OlvScripting",
"nInfComp_SeparationLevel" );

nSeparationLevel = sParameterContent.length > 0

   ? Number( sParameterContent )

   : -1; // default
```

6.10 GetPage

Get OlvPageCom object that corresponds to the document’s page.

6.10.1 Prototype:

```
GetPage(nPage : Integer) : OlvPageCom
```

6.10.2 Parameters:

nPage - integer number of page in Document. Page numeration is 0-based.

6.10.3 Return value:

OlvPageCom object

6.10.4 Example:

```
Var page = OlvDoc.GetPage(0);
```

6.11 GetIndexesOfTOCPages : VBArray

Get numbers of all pages containing “TOC_PAGE” property in their PropertySet

6.11.1 Prototype:

```
GetIndexesOfTOCPages() : OlvPageCom
```

6.11.2 Parameters:

6.11.3 Return value:

OlvPageCom object

6.11.4 Example:

```
var page = OlvDoc.GetPage(0);

page.SetProperty("TOC_PAGE", true, 0);

var HasToc = OlvDoc.HasTOCPages; // = true

var arrIndices = toArray(OlvDoc.GetIndexesOfTOCPages()); var Num =
arrIndices.length; // =1
```

6.12 GetBlockID : Integer

Unique ID for each block in the document.

Can be used to check if two OlvBlockCom objects are the same.

6.12.1 Prototype:

```
GetBlockID(iPageNum : Integer, indexBlock : Integer, typeBlock :
Integer) : Integer
```

6.12.2 Parameters:

iPageNum - 0-based number of Page in Document

indexBlock **-** 0-based index of block in a relevant (depending on block type) array of blocks (TextBlocks array, Horizontal Separators arrays etc.) on page.

typeBlock

Integer type.

Can be one of following values :

1 – TextBlock

2 – GraphBlock

4 - VerSeparator,

8 – HorSeparator

16 - TableBlock

These values are defined in XMD4Include.js by variables named "OlvBlockTypeXxxx"

6.12.3 Return value:

Integer ID for a Block or nul if the Block is not found in Page.

6.12.4 Example:

6.13 GetBlockByID – not used by the script

Look for Block with the specified ID among documents

6.13.1 Prototype:

```
GetBlockByID(blockID : Integer):OlvBlockCom
```

6.13.2 Parameters:

blockID - ID of Block. Should be number returned by GetBlockID method

6.13.3 Return value:

OlvBlockCom object or null.

6.13.4 Example:

6.14 RevisePictureRelatedBlocks

Revise (correct/improve) picture-related BSBlocks, such as image, picture-text, and picture-caption.

6.14.1 Prototype:

```
RevisePictureRelatedBlocks() : Boolean
```

6.14.2 Parameters:

6.14.3 Return value:

'**True**' upon completing the action

'**False**' when profile parameter '**RevisePictureBSBlocksMode**' does not allow, or the action has already completed

6.14.4 Example:

```
var bOK = OlvDoc.RevisePictureRelatedBlocks();
```

7. OlvPageCom

OlvPageCom – This is container object that holds collections of OlvTextBlockCom, OlvGraphBlockCom, OlvTableBlockCom, OlvVSeparatorCom and OlvHSeparatorCom.

The script must use **OlvDocCom.GetPage()** method to get a Page that cannot be created by the Script.

7.1.1 Derived from:

OlvPropertySet, OlvTextStyleGallery

7.1.2 Creation:

7.2 PROPERTIES AND METHODS

7.2.1 Properties Read/Write:

7.2.2 Properties Read only:

7.3 PageNumber : Integer

Number of Page in Doc (0-based)

7.3.1 Example:

See GetColumnRange

7.4 PageBox : OlvFRectCom

Page Box.

7.4.1 Example:

See GetColumnRange

7.5 ColumnsNumber: Integer

Number of columns on Page. Valid value is >=1 or -1.

If a Page has no "Column" layout then the value is -1

7.5.1 Example:

See GetColumnRange

7.6 TextBlocksNumber: Integer

Counts the number of TextBlocks on the Page.

7.6.1 Example:

See GetAllTextBlocks

7.7 GraphBlocksNumber: Integer

Count of GraphBlocks on Page.

7.7.1 Example:

See GetAllGraphBlocks

7.8 VSeparatorBlocksNumber: Integer

Count of VSeparators on Page

7.8.1 Example:

See GetAllVSeparatorBlocks

7.9 HSeparatorBlocksNumber

Count of HSeparators on Page

7.9.1 Example:

See GetAllHSeparatorBlocks

7.10 TableBlocksNumber

Count of TableBlocks on Page

7.10.1 Example:

See GetAllTableBlocks

7.10.2 Methods:

7.11 GetTextBlock : OlvTextBlockCom

Returns the i-th OlvTextBlockCom for Page

7.11.1 Prototype:

```
GetTextBlock(i : Integer) : OlvTextBlockCom
```

7.11.2 Parameters:

i - index of text block in collection of all text blocks in page

7.11.3 Return value:

OlvTextBlockCom object.

If i < 0 ot i >= number of text blocks in page then this method throws exception.

7.11.4 Example:

7.12 GetGraphBlock: OlvGraphBlockCom

Returns the i-th OlvGraphBlockCom for Page

7.12.1 Prototype:

```
GetGraphBlock(i : Integer) : OlvGraphBlockCom
```

7.12.2 Parameters:

i - index of graph block in collection of all graph blocks in page

7.12.3 Return value:

OlvGraphBlockCom object.

If i < 0 ot i >= number of graph blocks in page then this method throws exception.

7.12.4 Example:

7.13 GetHSeparatorBlock: OlvHSeparatorBlockCom

Returns the i-th OlvHSeparatorBlockCom for Page

7.13.1 Prototype:

```
GetHSeparatorBlock(i : Integer) : OlvHSeparatorBlockCom
```

7.13.2 Parameters:

i - index of horizontal separator in collection of all horizontal separators in page

7.13.3 Return value:

OlvHSeparatorBlockCom object.

If i < 0 or i >= number of horizontal separators in page then this method throws exception.

7.13.4 Example:

7.14 GetVSeparatorBlock: OlvVSeparatorBlockCom

Returns the i-th OlvVSeparatorBlockCom for Page

7.14.1 Prototype:

```
GetVSeparatorBlock(i : Integer) : OlvVSeparatorBlockCom
```

7.14.2 Parameters:

i - index of vertical separator in collection of all vertical separators in page

7.14.3 Return value:

OlvVSeparatorBlockCom object.

If i < 0 ot i >= number of vertical separators in page then this method throws exception.

7.14.4 Example:

7.15 GetTableBlock: OlvTableBlockCom

Returns the i-th OlvTableBlockCom for Page

7.15.1 Prototype:

```
GetTableBlock(i : Integer) : OlvTableBlockCom
```

7.15.2 Parameters:

i - index of Table block in collection of all table blocks in page

7.15.3 Return value:

OlvTableBlockCom object.

If i < 0 ot i >= number of table blocks in page then this method throws exception.

7.15.4 Example:

```
var count = Page.TableBlocksNumber;

var arr = toArray(Page.GetAllTableBlocks());

var len = arr.length; // count
```

7.16 GetAllTextBlocks: VBArray

Returns VBArray of all OlvTextBlockCom objects on Page

7.16.1 Prototype:

```
GetAllTextBlocks() : VBArray
```

7.16.2 Parameters:

7.16.3 Return value:

VBArray object. This array contains as its elements all text blocks in Page.

7.16.4 Example:

```
var count = Page.TextBlocksNumber;

var arr = toArray(Page.GetAllTextBlocks());
var len = arr.length; // count
```

7.17 GetAllGraphBlocks: VBArray

Returns VBArray of all OlvGraphBlockCom objects on Page

7.17.1 Prototype:

```
GetAllGraphBlocks() : VBArray
```

7.17.2 Parameters:

7.17.3 Return value:

VBArray object. This array contains as its elements all graph blocks in Page.

7.17.4 Example:

```
var count = Page.GraphBlocksNumber;

var arr = toArray(Page.GetAllGraphBlocks());

var len = arr.length; // count
```

7.18 GetAllHSeparatorBlocks: VBArray

Returns VBArray of all OlvHSeparatorBlockCom objects on Page

7.18.1 Prototype:

```
GetAllHSeparatorBlocks() : VBArray
```

7.18.2 Parameters:

7.18.3 Return value:

VBArray object. This array contains as its elements all horizontal separators in Page.

7.18.4 Example:

```
var count = Page.HSeparatorBlocksNumber;

var arr = toArray(Page.GetAllHSeparatorBlocks());

var len = arr.length; // count GetAllVSeparatorBlocks: VBArray
```

7.19 GetColumnRange: OlvRangeCom

Returns location of column on Page.

7.19.1 Prototype:

```
GetColumnRange(i: Integer) : OlvRangeCom
```

7.19.2 Parameters:

i - index of column on page

7.19.3 Return value:

Interval OlvRangeCom of left and right x-coordinates for the i-th column on Page. Value [-1,-1] means that no such column exist.

7.19.4 Example:

```
var Page_0 = OlvDoc.GetPage( 0 );

    var numPage = Page_0.PageNumber; // = 0

var Page_1 = OlvDoc.GetPage( 1 );

     // What page is wider?

if(Page_1.PageBox.Width() > Page_0. PageBox.Width())

{

OlvProgressIndicator.ReportLogText (“page 1 is wider than page 0”);

}

// Has page a column style?

var numColumns = Page_0.ColumnsNumber;

in(numColumns < 1)

{ // Page is not devided to Columns

OlvProgressIndicator.ReportLogText(“No Column Style.”);

}

Else

{// get left and right coordinate of the last

// column on page

var rangeLast = Page.GetColumnRange(numColumns - 1);

var left = rangeLast.begin;  // left x-coordinate of column
var right = rangeLast.end; //right x-coordinate of column
}
```

7.20 ComputeTextInBox: String

Returns text from rectangle on Page.

7.20.1 Prototype:

```
ComputeTextInBox(boxOnPage: OlvFRectCom) : String
```

7.20.2 Parameters:

boxOnPage - OlvFRect object defined rectangle on page

7.20.3 Return value:

String containing text written inside of rectangle on page. The text contains all words in there visual order. If Hebrew document it will be right-to-left order even for english text.

7.20.4 Example:

See ComputeBiDiTextInBox

7.21 ComputeBiDiTextInBox: String

Returns text from rectangle on Page. Text is ordered according to text direction defined in Profile.

7.21.1 Prototype:

```
ComputeBiDiTextInBox(boxOnPage: OlvFRectCom) : String
```

7.21.2 Parameters:

boxOnPage - OlvFRect object defined rectangle on page

7.21.3 Return value:

String containing text written inside of rectangle on page. The text contains all words in reading order. Hebrew text has right-to-left order of words. Mixed text contains control bidi characters around left-to-right substrings.

7.21.4 Example:

```
var TBlock = Page.GetTextBlock(0);
  // Get text of text block
var Text = TBlock.text;
// Get Box of bext block
var rBox = TBlock.BoxOnPage;
// Compute text inside of box. It is the same text
varTextTheSame = Page.ComputeTextInBox(rBox); // = Text
// Compute "ordered" text in box
// For LTR document it is the same as Text
  var TextBiDi = Page.ComputeBiDiTextInBox(rBox);
/*
```

For RTL document the result can be different if LTR text occur inside of box.

For example:

Text

בלפסט, 1920. יצא לאור מחדש באנגלית ע''י אוניברסיטת חיפה, of dyeing The ,Herzog Isaac rabbi Malacological Municipal :Society Malacological Israel Nahariya ,Israel ancient in purple

TextBiDi

בלפסט, &#x202D;.1920&#x202C; יצא לאור מחדש באנגלית ע''י אוניברסיטת חיפה, &#x202D;purple in ancient Israel, Nahariya Israel Malacological Society: Municipal Malacological rabbi Isaac Herzog, The dyeing of&#x202C;

8. OlvTagCom

Subset of sequential quads of OlvTextBlockCom object. Number of quads in OlvTagCom is >=1 and <= number of quads in text block.

8.1.1 Derived from:

OlvPropertySet

OlvTextStyleGallery

8.1.2 Creation:

OlvTagObject cann't be created explicitly but only using **CreateTag** method of OlvTextBlockCom object

8.2 PROPERTIES AND METHODS

8.2.1 Properties Read/Write:

8.2.2 Properties Read only:

8.3 StartLineIndex: Integer

Index of the Tag’s first text line in the collection of TextBlock’s lines.

8.4 EndLineIndex: Integer

Index of the Tag’s last text line in the collection of TextBlock’s lines.

8.5 StartQuadIndex: Integer

Index of the Tag’s first quad in the collection of first TextLine’s quads.

8.6 EndQuadIndex: Integer

Index of the Tag’s last quad in the collection of last TextLine’s quads.

8.6.1 Example:

```
Var Tag = Tblock.CreateTag(1, 0, 2, 3);
Var iL1 = Tag.StartLineIndex; // iL1 = 1
Var iL2 = Tag.EndLineIndex; // iL1 = 2
Var iQ1 = Tag.StartQuadIndex; // iQ1 = 0
Var iQ2 = Tag.EndLineIndex; // iQ1 = 3
```

8.6.2 Methods:

8.7 GetQuadBoxes: VBArray

8.7.1 Prototype:

```
GetQuadBoxes() : VBArray
```

8.7.2 Parameters:

8.7.3 Return value:

VBArray containg ColvFRect objects. Each rectangle is a box of a quad.

8.7.4 Example:

```
var boxArr = toArray(Tag.GetQuadBoxes());
```

8.7.5 Usage example:

```
Var TBlock = OlvDoc.GetPage(0).GetTextBlock(0);
// Create Tag starting from the first Quad of the second text line
// and ending in the last quad of the third line
Var indexStartLine = 1; // index of the second line
Var indexEndLine = 2; // index of the third line
Var indexStartQuad = 0;
Var tLineStart = Tblock.GetTextLine(indexStartLine);
Var tLineLast = Tblock.GetTextLine(indexEndLine);
Var indexLastQuad = tLineLast.QuadsNumber – 1;

// Create tag from quads of two text lines
Var Tag = Tblock.CreateTag(indexStartLine, indexStartQuad,
indexEndLine, indexLastQuad);

// Get boxes of all quads
 var boxArr = toArray(Tag.GetQuadBoxes());

// Result is always True
var bArrayOK = (boxArr.length == tLineLast.QuadsNumber +
tLineStart.QuadsNumber);
```

9. OlvBlockCom

This is a derived object that provides basic information and traversal functionality.

It is a base object for OlvTextBlockCom, OlvTableBlockCom, OlvGraphBlockCom, OlvHSeparatorCom and OlvVSeparatorCom objects.

It can contain the children blocks of any of the derived objects. There is a strictly geometrical relation between children and parent blocks: every children block is fully positioned in and enclosed by the parent’s rectangle, so the letter ‘G’ in properties and methods is for “GEOMETRICAL”.

There are no implemented restrictions for child/parent block type, but 'de facto' only two block objects OlvTableBlockCom and OlvGraphBlockCom may have child objects.

9.1.1 Derived from:

ElementProvider

OlvPropertySet

9.1.2 Creation:

n/a

9.2 PROPERTIES AND METHODS

9.2.1 Properties Read/Write:

n/a

9.2.2 Properties Read only:

9.3 Type : String

type of Block ("TextBlock", "GraphBlock" etc.)

refer to XMD4Include.js for full types list.

9.4 TypeNumeric : integer

Numeric enumerator of types (0x01 = TextBlock, 0x02 = GraphBlock etc.)

Refer to XMD4Include.js for full types list.

9.5 HasChildren : boolean

TRUE if current OlvBlockCom has "GEOMETRICAL" children BLOCKS

(OlvTableBlockCom containing OlvTextBlocksCom objects created from its cells and for OlvGraphBlockCom which is a frame for others).

9.6 NumberOfChildren : integer

number of "children" Blocks

9.7 HasGParent : boolean

TRUE if this block has “GEOMETRICAL” Parent block

9.7.1 Example:

See usage example below

9.7.2 Methods:

9.8 GetChildren

9.8.1 Description:

Retrieves “GEOMETRICAL” children of current Block.

9.8.2 Prototype:

```
function GetChildren() : Array;
```

9.8.3 Parameters:

n/a

9.8.4 Return value:

Array of Block objects that are its “GEOMETRICAL” children (one of OlvBlockCom’s derived objects).

9.8.5 Example:

See usage example below

9.9 GetGParent

9.9.1 Description:

Retrieves “GEOMETRICAL” parent of current Block if any.

9.9.2 Prototype:

```
function GetGParent() : OlvBlockCom’s derived object
```

9.9.3 Parameters:

9.9.4 Return value:

"GEOMETRICAL" Parent block (one of derived from OlvBlockCom objects) or "null".

9.9.5 Example:

See usage example below

9.10 Usage example:

```
// Block testing

// Type

AddMessage(NOTIFY, "Test OlvBlock::Type");

var bsType =  Block.Type;

if("TableBlock" == bsType)

         AddMessage(OK, "Type of block is \"TableBlock\"");

else

            AddMessage(FAIL, "Type of block is \"TableBlock\".
Returned - "+bsType);

AddMessage(NOTIFY, " ");


//TypeNumeric

AddMessage(NOTIFY, "Test OlvBlock::TypeNumeric");

var nType =  Block.TypeNumeric;


if(0x10 == nType)

      AddMessage(OK, "numType of block is 0x10");

else

      AddMessage(FAIL,  "numType of block is 0x10. Returned -
"+nType);

AddMessage(NOTIFY, " ");


// Has Children

AddMessage(NOTIFY, "Test OlvBlock::HasChildren");
```

```
if(true == Block.HasChildren)
{
         AddMessage(OK,"Returned -"+true);
      AddMessage(NOTIFY, " ");
      // Number of children
      AddMessage(NOTIFY, "Test OlvBlock::NumberOfChildren");
      var NumCh = Block.NumberOfChildren;

      if(NumCh != 58)
            AddMessage(FAIL,  "Number of children = 58. Returned -
"+NumCh);
      else
            AddMessage(OK,"Number of children = 58");
      AddMessage(NOTIFY, " ");
      //Get Children
      AddMessage(NOTIFY, "Test OlvBlock::GetChildren()");
      var ChArray = toArray(Block.GetChildren());

      if(ChArray == null)
            AddMessage(FAIL,  "Returned - " + ChArray);
      else
      {
                     AddMessage(OK, "Returned - !null");
            AddMessage(NOTIFY, " ");
                        //Has Parent
            AddMessage(NOTIFY, "Test OlvBlock::HasGParent");
                        var Child = ChArray[0];

            if(true == Child.HasGParent)
            {
                                 AddMessage(OK, "Returned - "+true);
                  AddMessage(NOTIFY, " ");
                  // Get Parent
                  AddMessage(NOTIFY, "Test OlvBlock::GetGParent()");
                  var Partnt = Child.GetGParent();


                  if(Partnt != null)
                     {
                                                      AddMessage(OK,
"Returned - !null");
                        AddMessage(NOTIFY, " ");
                     }
                        else
                     {
                        AddMessage(FAIL,  "Returned - " +
Partnt);
                        AddMessage(NOTIFY, " ");
                     }
               }
         else
      {
                                    AddMessage(FAIL,  "Returned
- " + false);
                     AddMessage(NOTIFY, " ");
                  }
         }
          }
```

10. OlvGraphBlockCom

This block is a derived object.

It represents an raster image or vector graphics block on the page.

It may serve as container frame for enclosed OlvGraphBlockCom and OlvTextBlockCom objects.

10.1.1 Derived from:

OlvBlockCom

10.1.2 Creation:

n/a

10.2 PROPERTIES AND METHODS

10.2.1 Properties Read/Write:

n/a

10.2.2 Properties Read only:

10.3 IsImage : boolean

10.3.1 Example:

```
// Testing IOlvGraphBlockCom

//IsImage

AddMessage(NOTIFY, "Test OlvGraphBlockCom::IsImage");

if(GrBlock.IsImage == true)

      AddMessage(OK,"Returned true");

else

      AddMessage(FAIL,"Returned false");

AddMessage(NOTIFY, " ");
```

10.3.2 Methods

n/a

11. OlvTableBlockCom

This is a derived object.

It represents a table block on the page.

May have children OlvTextBlocksCom objects that are created inside its cells.

11.1.1 Derived from:

OlvBlockCom

11.1.2 Creation:

n/a

11.2 PROPERTIES AND METHODS

11.2.1 Properties Read/Write:

n/a

11.2.2 Properties Read only:

n/a

11.2.3 Methods:

n/a

11.2.4 Usage Example:

```
var Block = Page1.GetTableBlock(0);
if (!Block)
      return;
AddMessage(NOTIFY, " ");
TypeNumeric : integer

//  Block testing

// Type

AddMessage(NOTIFY, "Test OlvBlock::Type");
var bsType =  Block.Type;
if("TableBlock" == bsType)
      AddMessage(OK, "Type of block is \"TableBlock\"");
else
      AddMessage(FAIL, "Type of block is \"TableBlock\". Returned -
"+bsType);
```

12. OlvHSeparatorBlockCom

This is a derived object that represents a horizontal separation block on the page.

12.1.1 Derived from:

OlvBlockCom

12.1.2 Creation:

n/a

12.2 PROPERTIES AND METHODS

12.2.1 Properties Read/Write:

n/a

12.2.2 Properties Read only:

n/a

12.2.3 Method:

n/a

12.2.4 Usage Example:

```
AddMessage(NOTIFY, "Calling OlvPageCom::GetHSeparatorBlock(0)");
var HSBlock = Page0.GetHSeparatorBlock(0);
if (!HSBlock)
   return;
```

13. OlvVSeparatorBlockCom

This is a derived object that represents vertical separation block on page.

13.1.1 Derived from:

OlvBlockCom

13.1.2 Creation:

n/a

13.2 PROPERTIES AND METHODS

13.2.1 Properties Read/Write:

n/a

13.2.2 Properties Read only:

n/a

13.2.3 Method:

n/a

13.2.4 Usage Example:

```
AddMessage(NOTIFY, "Calling OlvPageCom::GetVSeparatorBlock(0)");
var VSBlock = Page0.GetVSeparatorBlock(0);
if (!VSBlock)
            return;
```

14. OlvTextLineCom

Container for OlvQuadCom objects

14.1.1 Derived from:

ElementProvider

OlvTextStyleGallery

14.1.2 Creation:

n/a

14.2 PROPERTIES AND METHODS

14.2.1 Properties Read/Write:

n/a

14.2.2 Properties Read only:

14.3 Text : String

text of OlvTextLineCom

14.4 QuadsNumber : integer

Count the number of OlvQuadCom objects in OlvTextLineCom

14.4.1 Example:

See usage example below

14.4.2 Methods:

GetQuad(i) – return the i-th Quad (OlvQuadCom object).

GetQuads() – Return array of all Quads (OlvQuadCom objects).

14.5 GetQuad

14.5.1 Prototype:

function GetQuad(index : integer) : OlvQuadCom;

14.5.2 Parameters:

index : integer

index of desired OlvQuadCom

14.5.3 Return Value:

Indexed OlvQuadCom or "null" if out of bounds

14.5.4 Example:

See usage example below

14.6 GetQuads

14.6.1 Prototype:

function GetQuads() : Array of OlvQuadCom;

14.6.2 Parameters:

n/a

14.6.3 Return Value:

All OlvQuadCom objects of OlvTextLineCom

14.6.4 Example:

See usage example below

14.7 Usage Examples:

```
//Textline  testing
//Text
AddMessage(NOTIFY, "Test OlvTextLineCom::Text");
var bsText = TextLine.Text;
if(bsText == "Interactive Information Extraction")
      AddMessage(OK, "Text - \'"+bsText+"\'");
else
         AddMessage(FAIL, "Text - \'Interactive Information
Extraction\'. Returned - \'"+bsText+"\'");
AddMessage(NOTIFY, " ");


//QuadsNumber
AddMessage(NOTIFY, "Test OlvTextLineCom::QuadsNumber");
var QuadsNum = TextLine.QuadsNumber;
if(QuadsNum == 3)
         AddMessage(OK, "Quads number - "+QuadsNum);
else
         AddMessage(FAIL, "Quads number - 3. Returned -"+QuadsNum);
AddMessage(NOTIFY, " ");


//GetQuad
AddMessage(NOTIFY, "Test OlvTextLineCom::GetQuad(0)");
var quad = TextLine.GetQuad(0);
```

```
if(quad != null)
         AddMessage(OK, "Quads != null");
else
         AddMessage(FAIL, "Quads == null");
AddMessage(NOTIFY, " ");


//GetQuads
AddMessage(NOTIFY, "Test OlvTextLineCom::GetQuads()");
var qArr = toArray(TextLine.GetQuads());
if(qArr != null)
{
         if(qArr.length == 3)
                     AddMessage(OK, "length of quads array = 3");
         else
                     AddMessage(FAIL, "length of quads array = 3.
Returned "+qArr.length);
}
else
         AddMessage(FAIL, "quad array == null");
```

15. OlvQuadCom

Object usually describes a single word or part of a word in text

Sequential string of non-white space characters, having the same text format (style, size, color etc') and on the same line

One word may be split into a few Quads are when parts of that word have different text formats or when the word is split between lines

15.1.1 Derived from:

ElementProvider

15.1.2 Creation:

You can get a meaningful OlvQuadCom only as an output of some functions. You can not build a meaningful one by yourself, since it has no writeable properties.

15.2 PROPERTIES AND METHODS

15.2.1 Properties Read/Write:

15.2.2 Properties Read only:

15.3 Text: string

The text of Quad

15.4 FontFlags : number

A summation of numeric values which represent combination of the following string values/properties: “scanned”, “bold”, “italic” and “colored”

15.5 FontName : string

The name of the font

15.6 FontSize : float

The size of the font

15.6.1 Example:

```
var sText = Quad.Text;

var wFontFlags = Quad.FontFlags;

var sFontName  = Quad.FontName;

var f FontSize = Quad.FontSize;
```

15.6.2 Methods:

16. OlvBlockFilterCom

- Object that functions to "filter" blocks (OlvBlockCom) according to specific criteria. It can distinguish between blocks that match the criteria and those who don't (see methods).
- All criteria are composed of this objects' properties and inherited OlvPropertySet properties, and each property composes at least one criterion (detailed under each property).
- Most criteria have invalid values, which are ignored (same as blocking all blocks that match the criteria).
- A block matches the filter when it matches all the valid criteria. Any valid criteria that do not match, then the block does not match the filter (mismatch).

16.1.1 Derived from:

ElementProvider

OlvPropertySet

16.1.2 Creation:

```
var BlockFilter= new
ActiveXObject("OlvComWrapper.OlvBlockFilterCom");
```

16.2 PROPERTIES AND METHODS

16.2.1 Properties Read/Write:

16.3 FilterMatchCriteria: bollean

Any block either passes through or stops in this filter (OlvBlockFilterCom)

If **TRUE**, then only matching blocks pass

If **FALSE**, then only mismatched blocks pass

Default value = TRUE

See: match and mismatch definitions in class description.

See: all methods for usage of this property.

16.4 Box: OlvFRectCom

Defines a rectangle area in a (non-specific) page.

The rectangle's default (invalid) values = left = right = top = bottom = -1

Block-matching ignores this criterion when this property or LocationRelations property or block’s Box are invalid.

A matching block is a block located at the top of the page when this Box is located at bottom of the page and the LocationRelations==OlvLocAbove.

Formaly: A block matches when the location-relation of its Box to this Box is stated by the LocationRelations property.

See: LocationRelations, OrderRegime, OlvBlockCom.Box.

16.5 LocationRelations: integer

A summing (combination) of flags (integers), each defining one relationship between two rectangles on a (non-specific) page.

Possible location-relations are: Above, below, left of, right of, intersect, inside, outside

For all the possible location-relationships and their related flags see XMD4Include.js

Default (invalid) value = OlvLocUndefined.

This property is used only together with another property that can be related, such as Box.

See: Box.

16.6 OrderRelations: integer

A number defining one of the possible order relationships {before, after, overlap, inside}. For all the possible order relationships and their related integers see XMD4Include.js

Default (invalid) value = OlvOrdUndefined.

This property is used only with another property that can be related, such as PageNo, PagesRange or ColumnsInfo.

See: PageNo, ColumnsInfo, OrderRegime, PagesRange.

16.7 PageNo: integer

Page number

Default (invalid) value = -1.

Block-matching ignores this criterion when any of OrderRelations property or this PageNo value or the block’s PageNo value is invalid.

A block matches this criterion when the relationship of its PageNo value to this PageNo value is as stated by the OrderRelations property.

Note that OrderRelations ‘overlap’ and ‘inside’ both have the same meaning here: block.PageNo == this PageNo.

See: OrderRelations, OlvBlockCom.PageNo.

16.8 ColumnsInfo: OlvRangeCom

Range of text columns in a text page

Default (invalid) value is a total negative range (from= -1 to= -1).

Block-matching ignores this criterion when any of OrderRelations property or this ColumnsInfo or block’s ColumnsInfo is an invalid value.

Block matches this criterion when the relationship of block.ColumnsInfo to this ColumnsInfo is as stated by the OrderRelations property.

See: OrderRelations, OlvBlockCom.ColumnsInfo.

16.9 OrderRegime: integer

Number defining an order relations regime e.g. it defines an order between locations in a document. Possible regimes are :{ Reading Order, by PageNo and by Box.Top }. For all the possible order regimes and their related integers see XMD4Include.js

Default (invalid) value = OlvUndefinedOrder.

Block-matching ignores this criterion when any of OrderRegime or OrderRelations or this Box or block’s Box is invalid.

Example: If OrderRegime == OlvReadingOrder and OrderRelations ==OlvOrdBefore and this Box property is located at a page bottom then, a Block located at same page top matches this criterion.

Formaly: Block matches this criterion when, the order-relation under this OrderRegime of block.Box to this Box property is as stated by the OrderRelations property.

See: Box, OrderRelations, OlvBlockCom.ColumnsInfo, OlvBlockCom.Box.

16.10 BlockType : integer

A summation of flags (integers), each defining one block-type.

Possible types are: Text, Graph, Table, Separator…

For all the possible types and their related flags see XMD4Include.js

A block matches this criterion when its type is one of those defined here.

Default (invalid) value = OlvBlockTypeUnknown.

See: OlvBlockCom.TypeNumeric.

16.11 IgnoreRootBlocks: boolean

Default (invalid) value = false.

This criterion is ignored when it is 'false'.

A block matches this criterion when block.HasGParent == true.

See: IgnoreChildBlocks and OlvBlockCom.HasGParent.

16.12 IgnoreChildBlocks: boolean

Default (invalid) value = false.

This criterion is ignored when it is 'false'.

A block matches this criterion when block.HasGParent == false.

See: IgnoreRootBlocks and OlvBlockCom.HasGParent.

16.13 PagesRange: OlvRangeCom

Range of page numbers.

Default (invalid) value has a negative range (from= -1).

Block-matching ignores this criterion when any of PagesRange or OrderRelations or block.PageNo is invalid.

A block matches this criterion when, the relation of block.PageNo to this PagesRange is as stated by the OrderRelations property.

See: OrderRelations, OlvBlockCom.PageNo.

16.14 PageStatus: integer (for future use)

Number defining one of 3 possible statuses :{ single, left, right }.

For all the possible statuses and their related integers see XMD4Include.js

Default (invalid) value = OlvUndefinedPageStatus.

This criterion is ignored when it is invalid.

Currently when "single" then all blocks are matched, else all blocks are not matched

In future, to match the criterion, blocks will have to be in pages that have the same PageStatus.

16.15 PageLabel: string (for future use)

Currently not implemented.

In future, to match the criterion, blocks will have to be in pages that have the same PageLabels.

16.16 Inherited properties from OlvPropertySet

Inherited OlvPropertySet object composes it own matching criterion.

Default OlvPropertySet (invalid criterion) is an empty one e.g. (CountProperties() == 0).

Block matches this criterion when all properties of this OlvPropertySet exist "by name" in the block's OlvPropertySet. "by name" means that a property with that name exist even if its contents/ values are different.

See : OlvPropertySet class.

16.16.1 Properties Read only:

16.16.2 Example:

See file Test_BlockFilter.js

16.16.3 Methods:

16.17 PassBlock

Checks if a given OlvBlockCom 'passes' the current filter.

See FilterMatchCriteria property for more details.

16.17.1 Prototype:

```
PassBlock( Block : OlvBlockCom ) : boolean
```

16.17.2 Parameters:

Block : OlvBlockCom

16.17.3 Return value:

Boolean: 'true' when the Block passes this filter, else return 'false'.

See FilterMatchCriteria property for more details on 'passes'.

16.17.4 Example:

See file Test_BlockFilter.js

16.18 PassPage

Checks if a given OlvPageCom ‘passes’ current filter.

See ‘Return value’ for more details.

16.18.1 Prototype:

```
PassPage( Page : OlvPageCom) : boolean
```

16.18.2 Parameters:

Page: OlvPageCom

16.18.3 Return Value:

Boolean:

Returns '**true**' if the page ‘matches’ the PagesRange criterion when (FilterMatchCriteria == true) or when the criterion is ignored

else it returns 'false' when it ‘mismaches’ (FilterMatchCriteria == false).

16.18.4 Example:

See file Test_BlockFilter.js

16.19 PassBlockNoPageFiltering

Checks if a given OlvBlockCom ‘passes’ the current filter, while ignoring the PagesRange criterion.

See ‘return value’ more details.

16.19.1 Prototype:

```
PassBlock( Block : OlvBlockCom ) : boolean
```

16.19.2 Parameters:

Block - OlvBlockCom

16.19.3 Return value:

Boolean: ‘true’ when the Block ‘passes’ this filter, while ignoring PagesRange criterion, else return ‘false’.

See FilterMatchCriteria property for more details on ‘passes’.

16.19.4 Example:

See file Test_BlockFilter.js

16.20 FilterBlocks

‘Filters’ a group of blocks (OlvBlockCom) using the internal criteria.

16.20.1 Prototype:

```
FilterBlocks( InBlocks : VBArray, bFilterPages : boolean ) : VBArray
```

16.20.2 Parameters:

InBlocks - VBArray of the OlvBlockCom blocks to be filtered.

bFilterPages - Boolean. ‘True’ means to use PagesRange criterion, and ‘false’ means to ignore it.

16.20.3 Return Value:

A VBArray of all the OlvBlockCom blocks (of InBlocks) that ‘passed’ this current filter.

See FilterMatchCriteria property for more details on ‘passed’.

16.20.4 Example:

See file Test_BlockFilter.js

17. OlvBlockSortingCom

This object sorts a group of Blocks (OlvBlockCom) according to their reading order.

It has no properties.

17.1.1 Derived from:

17.1.2 Creation:

```
var sorter = new ActiveXObject("OlvComWrapper.OlvBlockSortingCom");
```

17.2 PROPERTIES AND METHODS

17.2.1 Properties Read/Write:

17.2.2 Properties Read only:

17.2.3 Methods:

17.3 SortBlocks

Sorts a group of Blocks (OlvBlockCom) according to a given order.

17.3.1 Prototype:

```
SortBlocks( ArrayIn : VBArray, SortingOrder : integer ) : VBArray
```

17.3.2 Parameters:

ArrayIn - VBArray can be created by toSafeArray() - in XMD4Include.js

SortingOrder - Integer, see XMD4Include.js for possible values

17.3.3 Return value:

A VBArray of the sorted Blocks (OlvBlockCom)

17.3.4 Example:

```
//========================================================
// This code finds the 1st text line of the 1st text block, in a
   reading order, from given 3 text blocks
//========================================================
// Create an Array from the given 3 text blocks
var BlockArray = new Array();
BlockArray.push( TextBlock1 );
BlockArray.push( TextBlock2 );
BlockArray.push( TextBlock3 );
// Convert the array into a COM safeArray
var safeArrayIn = toSafeArray( BlockArray ); // defined in
   XMD4Include.js
// Create a new OlvBlockSortingCom
var sorter = new ActiveXObject("OlvComWrapper.OlvBlockSortingCom");
// Sort the blocks by reading order
var safeArrayOut = sorter.SortBlocks( safeArrayIn, OlvReadingOrder );
// Convert the results into Array of Blocks (OlvBlockCom)
var arrSortedTBlocks = toArray( safeArrayOut ); // defined in
   XMD4Include.js
// Get the 1st text line of the 1st text block
var TextLine = arrSortedTBlocks[0].GetTextLine(0);
```

18. OlvFPointCom

This object defines the **x-** and **y-** coordinates of a point. Coordinate values are **FLOAT**.

18.1.1 Derived from:

18.1.2 Creation:

```
var point = new ActiveXObject("OlvComWrapper.OlvPointCom");
```

18.2 PROPERTIES AND METHODS

18.2.1 Properties Read/Write:

x - Specifies the point's x-coordinate float value

y - Specifies the point's y-coordinate float value

18.2.2 Example:

```
// Create point
      var Point = new ActiveXObject("OlvComWrapper.OlvFPointCom")
   ;
// Set values of coordinates
      Point.x = 2.5;
      Point.y = 3.5;
// Get values
      var x = Point.x;
      var y = Point.y;
```

18.2.3 Properties Read only:

18.2.4 Methods:

18.3 Offset

Defines the values of the x and y properties.

18.3.1 Prototype:

```
Offset( xOffset: Number, yOffset: Number)
```

18.3.2 Parameters:

xOffset - specifies the amount to offset of the x property.

yOffset - Specifies the amount to offset of the y property.

18.3.3 Return value:

18.3.4 Example:

```
// Create point

var Point = new ActiveXObject("OlvComWrapper.OlvFPointCom")    ;

// Set values of coordinates

Point.x = 2.5;

Point.y = 3.5;

// Offset

Point.Offset(2.0, 4.0)

// Get values

var x = Point.x;    // x = 4.5

var y = Point.y;    // y = 7.5
```

18.4 IsEqual

This method checks if two points are equal.

18.4.1 Prototype:

```
IsEqual(otherPoint: OlvFPointCom ): Boolean
```

18.4.2 Parameters:

otherPoint - specifies another point for comparison

18.4.3 Return value:

True if the points are equal; otherwise false.

18.4.4 Example:

```
var Point = new ActiveXObject("OlvComWrapper.OlvFPointCom")    ;

var Point2 = new ActiveXObject("OlvComWrapper.OlvFPointCom")   ;

// Set values of coordinates

Point.x = 1.5;  Point.y = 2.5; Point.Offset(1, 1);

Point2.x = 2.5; Point2.y = 3.5;

var Equal = Point.IsEqual(Point2); // Equal = true
```

18.5 Add

Adds values to the x and y properties.

18.5.1 Prototype:

```
Add(otherPoint: OlvFPointCom  )
```

18.5.2 Parameters:

otherPoint - specifies the amount to offset.

18.5.3 Return value:

18.5.4 Example:

See Sub;

18.6 Sub

Subtract values from the x and y properties.

18.6.1 Prototype:

```
Sub(other Point: OlvFPointCom  )
```

18.6.2 Parameters:

otherPoint - specifies the amount to subtract.

18.6.3 Return value:

18.6.4 Example:

```
var Point = new ActiveXObject("OlvComWrapper.OlvFPointCom")    ;

var Point2 = new ActiveXObject("OlvComWrapper.OlvFPointCom")   ;

// Set values of coordinates

Point.x = 15;  Point.y = 25;

Point2.x = 20; Point2.y = 30;

// Add

Point.Add(Point2);

var x = Point.x;  // x = 35
var y = Point.y;  // y = 55

// Subtract

Point2.Sub(Point);

var x = Point2.x;  // x = 5

var y = Point2.y;  // y = 5
```

19. OlvFuzzyCompareCom

- An object that compares between 2 other objects, where the quantities being compared are fuzzy.
- The objects that are compared are of types {float, OlvRangeCom, OlvFPointCom, OlvFRectCom }.
- Each comparison is done by calling one of the object's functions. The function returns a float with a special meaning that we'll call a "*Confidence*".
- **Confidence** is a float value in the range [0, 1] describing the certainty that a certain condition is fulfilled.

19.1.1 For example:

- Confidence 0.0 means "certainly not" (the condition is totally not fulfilled)
- Confidence 1.0 means "certainly yes" (the condition is totally fulfilled)
- Confidence 0.5 means "yes or no with equal confidence" (condition is half way between fulfilled and not fulfilled)

The amount of fuzziness is quantified by the object's properties below.

19.1.2 Derived from:

19.1.3 Creation:

```
var fzy = new ActiveXObject("OlvComWrapper.OlvFuzzyCompareCom");
```

19.2 PROPERTIES AND METHODS

19.2.1 Properties Read/Write:

The amount of fuzziness is quantified by the object's properties below. The Delta* properties are float values that can be either absolute or relative. **Absolute** delta means that, each of the numbers to bei compared is accurate up to +/-(delta/2). Respectively, same delta referring to range, point(x,y), or rectangle means that, each of their properties is accurate up to +/-(delta/2). **Relative** delta means that, each number's accuracy is +/-(number * delta).

Delta < 0 has the same meaning as Delta=0 which is the same as having no fuzziness (no delta).

19.3 Delta: float

For 1-dimension (float, OlvRangeCom) comparisons.

19.4 DeltaX : float

For X values of 2-dimensions (OlvFPointCom, OlvFRectCom) comparisons.

19.5 DeltaY : float

For Y values of 2-dimensions (OlvFPointCom, OlvFRectCom) comparisons.

19.6 DeltasRelative : boolean

True means that all properties (Delta, DeltaX, DeltaY) are relative, and False means that all properties (Delta, DeltaX, DeltaY) are absolute values.

19.6.1 Properties Read only:

19.6.2 Example:

19.6.3 Methods – comparing Number to Number:

19.7 AreNumbersEqual

Are the 2 numbers equal.

19.7.1 Prototype:

function AreNumbersEqual(First :float, Second :float) :Confidence (defined at the top)

19.7.2 Parameters:

First - Float

Second - Float

19.7.3 Return value:

Confidence (defined at the top)

19.7.4 Example:

```
fzy.DeltasRelative = false; (here true or false gives the same
results)

Confidence = fzy.AreNumbersEqual( 110.0, 100.0 );  //  Confidence ==
0.0

Confidence = fzy.AreNumbersEqual( 108.0, 100.0 );  //  Confidence ==
0.2

Confidence = fzy.AreNumbersEqual( 105.0, 100.0 );  //  Confidence ==
0.5

Confidence = fzy.AreNumbersEqual( 101.0, 100.0 );  //  Confidence ==
0.9

Confidence = fzy.AreNumbersEqual( 100.0, 100.0 );  //  Confidence ==
1.0

Confidence = fzy.AreNumbersEqual(   99.0, 100.0 );  //  Confidence ==
0.9

Confidence = fzy.AreNumbersEqual(   95.0, 100.0 );  //  Confidence ==
0.5

Confidence = fzy.AreNumbersEqual(   92.0, 100.0 );  //  Confidence ==
0.2

Confidence = fzy.AreNumbersEqual(   90.0, 100.0 );  //  Confidence ==
0.0
```

19.8 IsNumberSmallerThan

Is the first number smaller then the second.

19.8.1 Prototype:

```
function IsNumberSmallerThan ( First :float, Second :float )
:Confidence (defined at the top)
```

19.8.2 Parameters:

First :float

Second :float

19.8.3 Return value:

Confidence (defined at the top)

19.8.4 Example:

```
fzy.Delta = 10.0;

fzy.DeltasRelative = false; (here true or false gives the same results)

Cnfdnc = fzy. IsNumberSmallerThan ( 110.0, 100.0 );  //  Cnfdnc == 0.0

Cnfdnc = fzy. IsNumberSmallerThan ( 108.0, 100.0 );  //  Cnfdnc == 0.1

Cnfdnc = fzy. IsNumberSmallerThan ( 105.0, 100.0 );  //  Cnfdnc == 0.25

Cnfdnc = fzy. IsNumberSmallerThan ( 102.0, 100.0 );  //  Cnfdnc == 0.4

Cnfdnc = fzy. IsNumberSmallerThan ( 100.0, 100.0 );  //  Cnfdnc == 0.5

Cnfdnc = fzy. IsNumberSmallerThan (   98.0, 100.0 );  //  Cnfdnc == 0.6

Cnfdnc = fzy. IsNumberSmallerThan (   95.0, 100.0 );  //  Cnfdnc == 0.75

Cnfdnc = fzy. IsNumberSmallerThan (   92.0, 100.0 );  //  Cnfdnc == 0.9

Cnfdnc = fzy. IsNumberSmallerThan (   90.0, 100.0 );  //  Cnfdnc == 1.0
```

19.9 IsNumberGreaterThan

Is the first number greater then the second.

19.9.1 Prototype:

```
function IsNumberGreater Than ( First :float, Second :float ) : Confidence
```

19.9.2 Parameters:

First - Float

Second - Float

19.9.3 Return value:

Confidence (defined at the top)

19.9.4 Example:

```
fzy.Delta = 10.0;

fzy.DeltasRelative = false; //here true or false gives the same
results

Cnfdnc = fzy. IsNumberGreater ( 110.0, 100.0 );  //  Cnfdnc == 0.0

Cnfdnc = fzy. IsNumberGreater ( 108.0, 100.0 );  //  Cnfdnc == 0.1

Cnfdnc = fzy. IsNumberGreater ( 105.0, 100.0 );  //  Cnfdnc == 0.25

Cnfdnc = fzy. IsNumberGreater ( 102.0, 100.0 );  //  Cnfdnc == 0.4

Cnfdnc = fzy. IsNumberGreater ( 100.0, 100.0 );  //  Cnfdnc == 0.5

Cnfdnc = fzy. IsNumberGreater (   98.0, 100.0 );  //  Cnfdnc == 0.6

Cnfdnc = fzy. IsNumberGreater (   95.0, 100.0 );  //  Cnfdnc == 0.75

Cnfdnc = fzy. IsNumberGreater (   92.0, 100.0 );  //  Cnfdnc == 0.9

Cnfdnc = fzy. IsNumberGreater (   90.0, 100.0 );  //  Cnfdnc == 1.0
```

19.9.5 Methods – comparing Number to Range:

19.10 IsNumberInsideRange

Is the number inside the range.

19.10.1 Prototype:

```
function IsNumberInsideRange ( Number :float, Range :OlvRangeCom )
:Confidence (defined at the top)
```

19.10.2 Parameters:

Number - Float

Range - OlvRangeCom

19.10.3 Return value:

Confidence (defined at the top)

19.10.4 Example:

```
fzy.DeltasRelative = false;

fzy.Delta = 20.0;

var range = new ActiveXObject("OlvComWrapper.OlvRangeCom");

range.from = 100.0;

range.to   = 300.0;

Cnfdnc = fzy.IsNumberInsideRange( 110.0, range ); // Cnfdnc = 0.75
```

19.11 IsNumberOutsideRange

Is the number outside the range.

19.11.1 Prototype:

```
function IsNumberOutsideRange ( Number :float, Range :OlvRangeCom )
:Confidence (defined at the top)
```

19.11.2 Parameters:

Number - Float

Range - OlvRangeCom

19.11.3 Return value:

Confidence (defined at the top)

19.11.4 Example:

```
fzy.DeltasRelative = false;

fzy.Delta = 20.0;

var range = new ActiveXObject("OlvComWrapper.OlvRangeCom");

range.from = 100.0;

range.to   = 300.0;

Cnfdnc = fzy.IsNumberOutsideRange( 100.0, range ); // Cnfdnc = 0.50
```

19.12 IsNumberBeforeRange

The number before the range

19.12.1 Prototype:

```
function IsNumberBeforeRange ( Number :float, Range :OlvRangeCom )
:Confidence (defined at the top)
```

19.12.2 Parameters:

Number - Float

Range - OlvRangeCom

19.12.3 Return value:

Confidence (defined at the top)

19.12.4 Example:

```
fzy.Delta = 20.0;

var range = new ActiveXObject("OlvComWrapper.OlvRangeCom");

range.from = 100.0;

range.to   = 300.0;

Cnfdnc = fzy.IsNumberBeforeRange( 110.0, range ); // Cnfdnc = 0.45
```

19.13 IsNumberAfterRange

Is the number after the range.

19.13.1 Prototype:

```
function IsNumberAfterRange ( Number :float, Range :OlvRangeCom )
:Confidence (defined at the top)
```

19.13.2 Parameters:

Number :float

Range :OlvRangeCom

19.13.3 Return value:

Confidence (defined at the top)

19.13.4 Example:

```
fzy.DeltasRelative = false;

fzy.Delta = 20.0;

var range = new ActiveXObject("OlvComWrapper.OlvRangeCom");

range.from = 100.0;

range.to   = 300.0;

Cnfdnc = fzy.IsNumberAfterRange( 32.0, range ); // Cnfdnc = 0.60
```

19.14 IsNumberAtRangeStar

Does the number equals the range starting location.

19.14.1 Prototype:

```
function IsNumberAtRangeStar ( Number :float, Range :OlvRangeCom )
:Confidence (defined at the top)
```

19.14.2 Parameters:

Number :float

Range :OlvRangeCom

19.14.3 Return value:

Confidence (defined at the top)

19.14.4 Example:

```
fzy.DeltasRelative = false;

fzy.Delta = 20.0;

range.from = 100.0;

range.to   = 300.0;

Cnfdnc = fzy. IsNumberAtRangeStar ( 125.0, range ); // Cnfdnc = 0.75
```

19.15 IsNumberAtRangeEnd

Does the number equal the range ending location

19.15.1 Prototype:

```
function IsNumberAtRangeEnd ( Number :float, Range :OlvRangeCom )
:Confidence (defined at the top)
```

19.15.2 Parameters:

Number :float

Range :OlvRangeCom

19.15.3 Return value:

Confidence (defined at the top)

19.15.4 Example:

```
fzy.DeltasRelative = false;

fzy.Delta = 20.0;

var range = new ActiveXObject("OlvComWrapper.OlvRangeCom");

range.from = 100.0;

range.to   = 300.0;

Cnfdnc = fzy. IsNumberAtRangeEnd ( 340.0, range ); // Cnfdnc = 0.60
```

19.15.5 Methods – comparing Range to Range:

19.16 IsRangeInsideOther

Is first range is totally inside the second range

19.16.1 Prototype:

```
function IsRangeInsideOther ( First : OlvRangeCom, Second
:OlvRangeCom ) :Confidence (defined at the top)
```

19.16.2 Parameters:

First : OlvRangeCom

Second : OlvRangeCom

19.16.3 Return value:

Confidence (defined at the top). It does not indicate the amount of overlap between the two. E.g. with 50% overlap the Confidence (return value) will probably be 0.0, while Confidence=0.50 will happen with almost 100% overlap.

19.16.4 Example:

```
fzy.DeltasRelative = false;

fzy.Delta = 20.0;

var First = new ActiveXObject("OlvComWrapper.OlvRangeCom");

var Second = new ActiveXObject("OlvComWrapper.OlvRangeCom");

First.from = 120.0;

First.to   = 340.0;

Second.from = 100.0;

Second.to   = 300.0;

Cnfdnc = fzy. IsRangeInsideOther (First, Second); // Cnfdnc = 0.30
```

19.17 IsRangeEncloseOther

Is second range is totally inside the first range (opposite to IsRangeInsideOther)

19.17.1 Prototype:

```
function IsRangeEncloseOther ( First : OlvRangeCom, Second
:OlvRangeCom ) :Confidence (defined at the top)
```

19.17.2 Parameters:

First : OlvRangeCom

Second : OlvRangeCom

19.17.3 Return value:

Confidence (defined at the top). It does not indicate the amount of overlap between the two. E.g. with 50% overlap the Confidence (return value) will probably be 0.0, while Confidence=0.50 will happen with almost 100% overlap.

19.17.4 Example:

```
fzy.DeltasRelative = false;

fzy.Delta = 20.0;

var First = new ActiveXObject("OlvComWrapper.OlvRangeCom");

var Second = new ActiveXObject("OlvComWrapper.OlvRangeCom");

First.from = 120.0;

First.to   = 340.0;

Second.from = 100.0;

Second.to   = 300.0;

Cnfdnc = fzy. IsRangeEncloseOther (First, Second); // Cnfdnc = 0.40

Cnfdnc = fzy. IsRangeEncloseOther (Second, First); // Cnfdnc = 0.30
```

19.18 IsRangeIntersectOther

19.18.1 Description:

Is first range Intersects the second range

19.18.2 Prototype:

```
function IsRangeIntersectOther ( First : OlvRangeCom, Second
:OlvRangeCom ) :Confidence (defined at the top)
```

19.18.3 Parameters:

First : OlvRangeCom

Second : OlvRangeCom

19.18.4 Return value:

Confidence (defined at the top). It does not indicate the amount of overlap between the two. E.g. with 50% overlap the Confidence (return value) will probably be 1.0, while Confidence=0.50 will happen with almost 0% overlap.

19.18.5 Example:

```
fzy.DeltasRelative = false;

fzy.Delta = 20.0;

var First = new ActiveXObject("OlvComWrapper.OlvRangeCom");
var Second = new ActiveXObject("OlvComWrapper.OlvRangeCom");

First.from = 340.0;

First.to   = 444.0;

Second.from = 100.0;

Second.to   = 300.0;

Cnfdnc = fzy. IsRangeIntersectOther(First, Second); // Cnfdnc = 0.30

First.from = 0.0;

First.to   = 120.0;

Cnfdnc = fzy. IsRangeIntersectOther(First, Second); // Cnfdnc = 0.60
```

19.19 IsRangeAfterOther

Is first range is after the second range

19.19.1 Prototype:

```
function IsRangeAfterOther ( First : OlvRangeCom, Second :OlvRangeCom
) :Confidence (defined at the top)
```

19.19.2 Parameters:

First : OlvRangeCom

Second : OlvRangeCom

19.19.3 Return value:

Confidence (defined at the top)

19.19.4 Example:

```
fzy.DeltasRelative = false;

fzy.Delta = 20.0;

var First = new ActiveXObject("OlvComWrapper.OlvRangeCom");

var Second = new ActiveXObject("OlvComWrapper.OlvRangeCom");

First.from = 340.0;

First.to   = 444.0;

Second.from = 100.0;

Second.to   = 300.0;

Cnfdnc = fzy. IsRangeAfterOther(First, Second); // Cnfdnc = 0.70
```

19.20 IsRangeBeforeOther

Is the first range before the second range

19.20.1 Prototype:

```
function IsRangeBeforeOther ( First : OlvRangeCom, Second
:OlvRangeCom ) :Confidence (defined at the top)
```

19.20.2 Parameters:

First : OlvRangeCom

Second : OlvRangeCom

19.20.3 Return value:

Confidence (defined at the top)

19.20.4 Example:

```
fzy.DeltasRelative = false;

fzy.Delta = 20.0;

var First = new ActiveXObject("OlvComWrapper.OlvRangeCom");

var Second = new ActiveXObject("OlvComWrapper.OlvRangeCom");

First.from = 0.0;

First.to   = 120.0;

Second.from = 100.0;

Second.to   = 300.0;

Cnfdnc = fzy. IsRangeBeforeOther (First, Second); // Cnfdnc = 0.40
```

19.20.5 Methods – comparing Point to Rectangle:

19.21 IsPointBelowRect

Is the point.y located lower than the rectangle's bottom

19.21.1 Prototype:

```
function IsPointBelowRect( fPoint: OlvFPointCom, fRect : OlvFRectCom)
:Confidence (defined at the top)
```

19.21.2 Parameters:

fPoint: OlvFPointCom

fRect : OlvFRectCom

19.21.3 Return value:

Confidence (defined at the top)

19.21.4 Example:

```
fzy.DeltasRelative = false;

fzy.DeltaX = 50.0;

fzy.DeltaY = 25.0;

var fPoint = new ActiveXObject("OlvComWrapper.OlvFPointCom");

fPoint.x = 320;

fPoint.y = 720;

var fRect = new ActiveXObject("OlvComWrapper.OlvFRectCom");

fRect.left = 100;

fRect.top = 900;

fRect.right = 300;

fRect.bottom = 700;

Cnfdnc = fzy.IsPointBelowRect( fPoint, fRect ); // Cnfdnc = 0.10
```

19.22 IsPointRightToRect

Is the point.x located right to rectangle's right border

19.22.1 Prototype:

```
function IsPointRightToRect ( fPoint: OlvFPointCom, fRect :
OlvFRectCom) :Confidence (defined at the top)
```

19.22.2 Parameters:

fPoint: OlvFPointCom

fRect : OlvFRectCom

19.22.3 Return value:

Confidence (defined at the top)

19.22.4 Example:

```
fzy.DeltasRelative = false;

fzy.DeltaX = 50.0;

fzy.DeltaY = 25.0;

var fPoint = new ActiveXObject("OlvComWrapper.OlvFPointCom");

fPoint.x = 320;

fPoint.y = 720;

var fRect = new ActiveXObject("OlvComWrapper.OlvFRectCom");

fRect.left = 100;

fRect.top = 900;

fRect.right = 300;

fRect.bottom = 700;

Cnfdnc = fzy.IsPointRightToRect( fPoint, fRect ); // Cnfdnc = 0.10
```

19.23 IsPointRightLowerToRect

Is the point.x located right to rectangle's right border, and point.y lower than rectangle's bottom.

19.23.1 Prototype:

```
function IsPointRightLowerToRect( fPoint: OlvFPointCom, fRect :
OlvFRectCom) :Confidence (defined at the top)
```

19.23.2 Parameters:

fPoint: OlvFPointCom

fRect : OlvFRectCom

19.23.3 Return value:

Confidence (defined at the top)

19.23.4 Example:

```
fzy.DeltasRelative = false;

fzy.DeltaX = 50.0;

fzy.DeltaY = 25.0;

var fPoint = new ActiveXObject("OlvComWrapper.OlvFPointCom");

fPoint.x = 320;

fPoint.y = 720;

var fRect = new ActiveXObject("OlvComWrapper.OlvFRectCom");

fRect.left = 100;

fRect.top = 900;

fRect.right = 300;

fRect.bottom = 700;

Cnfdnc = fzy.IsPointRightLowerToRect ( fPoint, fRect ); // Cnfdnc =
0.10
```

19.24 IsPointInsideRect

Is the point located inside the rectangle

19.24.1 Prototype:

```
function IsPointInsideRect( fPoint: OlvFPointCom, fRect :
OlvFRectCom) :Confidence (defined at the top)
```

19.24.2 Parameters:

fPoint: OlvFPointCom

fRect : OlvFRectCom

19.24.3 Return value:

Confidence (defined at the top)

19.24.4 Example:

```
fzy.DeltasRelative = false;
fzy.DeltaX = 50.0;

fzy.DeltaY = 25.0;

var fPoint = new ActiveXObject("OlvComWrapper.OlvFPointCom");

fPoint.x = 320;

fPoint.y = 720;

var fRect = new ActiveXObject("OlvComWrapper.OlvFRectCom");

fRect.left = 100;

fRect.top = 900;

fRect.right = 300;

fRect.bottom = 700;

Cnfdnc = fzy.IsPointInsideRect ( fPoint, fRect ); // Cnfdnc = 0.30

fPoint.x = 90;

fPoint.y = 890;

Cnfdnc = fzy.IsPointInsideRect ( fPoint, fRect ); // Cnfdnc = 0.40
```

19.25 IsPointOutsideRect

Is the point located Outside the rectangle

19.25.1 Prototype:

```
function IsPointOutsideRect( fPoint: OlvFPointCom, fRect :
OlvFRectCom) :Confidence (defined at the top)
```

19.25.2 Parameters:

fPoint: OlvFPointCom

fRect : OlvFRectCom

19.25.3 Return value:

Confidence (defined at the top)

19.25.4 Example:

```
fzy.DeltasRelative = false;

fzy.DeltaX = 50.0;

fzy.DeltaY = 25.0;

var fPoint = new ActiveXObject("OlvComWrapper.OlvFPointCom");
```

```
fPoint.x = 320;

fPoint.y = 720;

var fRect = new ActiveXObject("OlvComWrapper.OlvFRectCom");

fRect.left = 100;

fRect.top = 900;

fRect.right = 300;

fRect.bottom = 700;

Cnfdnc = fzy.IsPointOutsideRect ( fPoint, fRect ); // Cnfdnc = 0.70

fPoint.x = 90;

fPoint.y = 890;

Cnfdnc = fzy.IsPointOutsideRect ( fPoint, fRect ); // Cnfdnc = 0.60
```

19.26 IsPointCenterLowerToRect

Is point.x located between right and left rectangle's border, and point.y is lower than rectangle's bottom.

19.26.1 Prototype:

```
function IsPointCenterLowerToRect( fPoint: OlvFPointCom, fRect :
OlvFRectCom) :Confidence (defined at the top)
```

19.26.2 Parameters:

fPoint: OlvFPointCom

fRect : OlvFRectCom

19.26.3 Return value:

Confidence (defined at the top)

19.26.4 Example:

```
fzy.DeltasRelative = false;

fzy.DeltaX = 50.0;

fzy.DeltaY = 25.0;

var fPoint = new ActiveXObject("OlvComWrapper.OlvFPointCom");

fPoint.x = 320;

fPoint.y = 720;
```

```
var fRect = new ActiveXObject("OlvComWrapper.OlvFRectCom");

fRect.left = 100;

fRect.top = 900;

fRect.right = 300;

fRect.bottom = 700;

Cnfdnc = fzy.IsPointCenterLowerToRect( fPoint, fRect ); // Cnfdnc =
0.10
```

19.27 IsPointRightCenterToRect

Is point.x located right to rectangle's right border, and point.y is between rectangle's top and bottom.

19.27.1 Prototype:

```
function IsPointRightCenterToRect( fPoint: OlvFPointCom, fRect :
OlvFRectCom) :Confidence (defined at the top)
```

19.27.2 Parameters:

fPoint: OlvFPointCom fRect : OlvFRectCom Return value: Confidence (defined at the top)

19.27.3 Example:

```
fzy.DeltasRelative = false;

fzy.DeltaX = 50.0;

fzy.DeltaY = 25.0;

var fPoint = new ActiveXObject("OlvComWrapper.OlvFPointCom");

fPoint.x = 320;

fPoint.y = 720;

var fRect = new ActiveXObject("OlvComWrapper.OlvFRectCom");

fRect.left = 100;

fRect.top = 900;

fRect.right = 300;

fRect.bottom = 700;

Cnfdnc = fzy.IsPointRightCenterToRect( fPoint, fRect ); // Cnfdnc =
0.70
```

19.28 IsPointLeftToRect

Is point.x located left to the rectangle's left border

19.28.1 Prototype:

```
function IsPointLeftToRect( fPoint: OlvFPointCom, fRect :
OlvFRectCom) :Confidence (defined at the top)
```

19.28.2 Parameters:

fPoint: OlvFPointCom

fRect : OlvFRectCom

19.28.3 Return value:

Confidence (defined at the top)

19.28.4 Example:

```
fzy.DeltasRelative = false;

fzy.DeltaX = 50.0;

fzy.DeltaY = 25.0;

var fPoint = new ActiveXObject("OlvComWrapper.OlvFPointCom");

fPoint.x = 90;

fPoint.y = 890;

var fRect = new ActiveXObject("OlvComWrapper.OlvFRectCom");

fRect.left = 100;

fRect.top = 900;

fRect.right = 300;

fRect.bottom = 700;

Cnfdnc = fzy.IsPointLeftToRect( fPoint, fRect ); // Cnfdnc = 0.60
```

19.29 IsPointAboveRect

Is the point.y located above the rectangle's top

19.29.1 Prototype:

```
function IsPointAboveRect( fPoint: OlvFPointCom, fRect : OlvFRectCom)
:Confidence (defined at the top)
```

19.29.2 Parameters:

fPoint: OlvFPointCom

fRect : OlvFRectCom

19.29.3 Return value:

Confidence (defined at the top)

19.29.4 Example:

```
fzy.DeltasRelative = false;

fzy.DeltaX = 50.0;

fzy.DeltaY = 25.0;

var fPoint = new ActiveXObject("OlvComWrapper.OlvFPointCom");

fPoint.x = 90;

fPoint.y = 890;

var fRect = new ActiveXObject("OlvComWrapper.OlvFRectCom");

fRect.left = 100;

fRect.top = 900;

fRect.right = 300;

fRect.bottom = 700;

Cnfdnc = fzy.IsPointAboveRect( fPoint, fRect ); // Cnfdnc = 0.30
```

19.30 IsPointLeftUpperToRect

Is point.x located left to rectangle's left border, and point.y is above rectangle's top.

19.30.1 Prototype:

```
function IsPointLeftUpperToRect( fPoint: OlvFPointCom, fRect :
OlvFRectCom) :Confidence (defined at the top)
```

19.30.2 Parameters:

fPoint: OlvFPointCom

fRect : OlvFRectCom

19.30.3 Return value:

Confidence (defined at the top)

19.30.4 Example:

```
fzy.DeltasRelative = false;

fzy.DeltaX = 50.0;

fzy.DeltaY = 25.0;

var fPoint = new ActiveXObject("OlvComWrapper.OlvFPointCom");

    fPoint.x = 90;

fPoint.y = 890;

var fRect = new ActiveXObject("OlvComWrapper.OlvFRectCom");

fRect.left = 100;

fRect.top = 900;

fRect.right = 300;

fRect.bottom = 700;

Cnfdnc = fzy.IsPointLeftUpperToRect( fPoint, fRect ); // Cnfdnc =
0.30
```

19.31 IsPointCenterUpperToRect

Is point.x located between rectangle's left and right borders, and point.y is above rectangle's top.

19.31.1 Prototype:

```
function IsPointCenterUpperToRect( fPoint: OlvFPointCom, fRect :
OlvFRectCom) :Confidence (defined at the top)
```

19.31.2 Parameters:

fPoint: OlvFPointCom

fRect : OlvFRectCom

19.31.3 Return value:

Confidence (defined at the top)

19.31.4 Example:

```
fzy.DeltasRelative = false;

fzy.DeltaX = 50.0;

fzy.DeltaY = 25.0;

var fPoint = new ActiveXObject("OlvComWrapper.OlvFPointCom");

    fPoint.x = 90;

fPoint.y = 890;

var fRect = new ActiveXObject("OlvComWrapper.OlvFRectCom");

fRect.left = 100;

fRect.top = 900;

fRect.right = 300;

fRect.bottom = 700;

Cnfdnc = fzy.IsPointCenterUpperToRect( fPoint, fRect ); // Cnfdnc =
0.30
```

19.32 IsPointLeftCenterToRect

Is point.x located left to rectangle’s left border, and point.y is between rectangle’s top and bottom.

19.32.1 Prototype:

```
function IsPointLeftCenterToRect( fPoint: OlvFPointCom, fRect :
OlvFRectCom) :Confidence (defined at the top)
```

19.32.2 Parameters:

fPoint: OlvFPointCom

fRect : OlvFRectCom

19.32.3 Return value:

Confidence (defined at the top)

19.32.4 Example:

```
fzy.DeltasRelative = false;

fzy.DeltaX = 50.0;

fzy.DeltaY = 25.0;

var fPoint = new ActiveXObject("OlvComWrapper.OlvFPointCom");

fPoint.x = 90;

fPoint.y = 890;

var fRect = new ActiveXObject("OlvComWrapper.OlvFRectCom");

fRect.left = 100;

fRect.top = 900;

fRect.right = 300;

fRect.bottom = 700;

Cnfdnc = fzy.IsPointLeftCenterToRect( fPoint, fRect ); // Cnfdnc =
0.60
```

19.33 IsPointRightUpperToRect

Is point.x located right to rectangle's right border, and point.y is above rectangle's top.

19.33.1 Prototype:

```
function IsPointRightUpperToRect( fPoint: OlvFPointCom, fRect :
OlvFRectCom) :Confidence (defined at the top)
```

19.33.2 Parameters:

fPoint: OlvFPointCom

fRect : OlvFRectCom

19.33.3 Return value:

Confidence (defined at the top)

19.33.4 Example:

```
fzy.DeltasRelative = false;

fzy.DeltaX = 50.0;

fzy.DeltaY = 25.0;
```

```
var fPoint = new ActiveXObject("OlvComWrapper.OlvFPointCom");

    fPoint.x = 280;

fPoint.y =900;

var fRect = new ActiveXObject("OlvComWrapper.OlvFRectCom");

fRect.left = 100;

fRect.top = 900;

fRect.right = 300;

fRect.bottom = 700;

Cnfdnc = fzy.IsPointRightUpperToRect( fPoint, fRect ); // Cnfdnc =
0.30
```

19.34 IsPointLeftLowerToRect

Is point.x located left to rectangle's left border, and point.y is below rectangle's bottom.

19.34.1 Prototype:

```
function IsPointLeftLowerToRect( fPoint: OlvFPointCom, fRect :
OlvFRectCom) :Confidence (defined at the top)
```

19.34.2 Parameters:

fPoint: OlvFPointCom

fRect : OlvFRectCom

19.34.3 Return value:

Confidence (defined at the top)

19.34.4 Example:

```
fzy.DeltasRelative = false;

fzy.DeltaX = 50.0;

fzy.DeltaY = 25.0;

var fPoint = new ActiveXObject("OlvComWrapper.OlvFPointCom");

    fPoint.x = 110;

fPoint.y = 700;

var fRect = new ActiveXObject("OlvComWrapper.OlvFRectCom");
```

```
fRect.left = 100;

fRect.top = 900;

fRect.right = 300;

fRect.bottom = 700;

Cnfdnc = fzy.IsPointLeftLowerToRect( fPoint, fRect ); // Cnfdnc =
0.40
```

19.34.5 Methods – comparing Rectangle to Rectangle:

19.35 IsRectLeftToOther

Is first rectangle's right border located left to other rectangle's left border.

19.35.1 Prototype:

```
function IsRectLeftToOther( Rect: OlvFRectCom, Other : OlvFRectCom)
:Confidence (defined at the top)
```

19.35.2 Parameters:

Rect: OlvFRectCom

Other: OlvFRectCom

19.35.3 Return value:

Confidence (defined at the top). It does not indicate the amount of overlap between the two. E.g. with 50% overlap the Confidence (return value) will probably be 0.0, while Confidence=0.50 will happen with almost 0% overlap.

19.35.4 Example:

```
fzy.DeltasRelative = false;

fzy.DeltaX = 60.0;

fzy.DeltaY = 80.0;

var fRect = new ActiveXObject("OlvComWrapper.OlvFRectCom");

fRect.left = 100;

fRect.top = 900;

fRect.right = 300;

fRect.bottom = 700;

var rctLT = new ActiveXObject("OlvComWrapper.OlvFRectCom");
```

```
rctLT.left   = 0;

rctLT.top    = 1111;

rctLT.right   = 115;

rctLT.bottom = 916;

Cnfdnc = fzy.IsRectLeftToOther( fRect, rctRB ); // Cnfdnc = 0.666666

fzy.DeltaX = 0.3;

fzy.DeltaY = 0.4;

fzy.DeltasRelative = true;

var rctRB = new ActiveXObject("OlvComWrapper.OlvFRectCom");

rctRB.left   = 320;

rctRB.top    = 760;

rctRB.right   = 444;

rctRB.bottom = 666;

Cnfdnc = fzy.IsRectLeftToOther( rctLT, fRect ); // Cnfdnc = 0.375
```

19.36 IsRectRightToOther

Is first rectangle's left border located Right to Other rectangle's right border.

19.36.1 Prototype:

```
function IsRectRightToOther( Rect: OlvFRectCom, Other : OlvFRectCom)
:Confidence (defined at the top)
```

19.36.2 Parameters:

Rect: OlvFRectCom

Other: OlvFRectCom

19.36.3 Return value:

Confidence (defined at the top). It does not indicate the amount of overlap between the two. E.g. with 50% overlap the Confidence (return value) will probably be 0.0, while Confidence=0.50 will happen with almost 0% overlap.

19.36.4 Example:

```
  fzy.DeltasRelative = false;

fzy.DeltaX = 60.0;
fzy.DeltaY = 80.0;
var fRect = new ActiveXObject("OlvComWrapper.OlvFRectCom");
  fRect.left = 100;
fRect.top = 900;
fRect.right = 300;
fRect.bottom = 700;
var rctLT = new ActiveXObject("OlvComWrapper.OlvFRectCom");
rctLT.left   = 0;
rctLT.top    = 1111;
rctLT.right   = 115;
rctLT.bottom = 916;
Cnfdnc = fzy. IsRectRightToOther ( fRect, rctLT ); // Cnfdnc = 0.375
fzy.DeltaX = 0.3;
fzy.DeltaY = 0.4;
fzy.DeltasRelative = true;
var rctRB = new ActiveXObject("OlvComWrapper.OlvFRectCom");
rctRB.left   = 320;
rctRB.top    = 760;
rctRB.right   = 444;
rctRB.bottom = 666;
Cnfdnc = fzy.IsRectRightToOther ( rctRB, fRect ); // Cnfdnc =
0.666666
```

19.37 IsRectAboveOther

Is first rectangle’s bottom is above other rectangle’s top.

19.37.1 Prototype:

```
function IsRectAboveOther( Rect: OlvFRectCom, Other : OlvFRectCom)
:Confidence (defined at the top)
```

19.37.2 Parameters:

Rect: OlvFRectCom

Other: OlvFRectCom

19.37.3 Return value:

Confidence (defined at the top). It does not indicate the amount of overlap between the two. E.g. with 50% overlap the Confidence (return value) will probably be 0.0, while Confidence=0.50 will happen with almost 0% overlap.

19.37.4 Example:

```
fzy.DeltaX = 0.3;
fzy.DeltaY = 0.4;
fzy.DeltasRelative = true;
var rctLT = new ActiveXObject("OlvComWrapper.OlvFRectCom");
rctLT.left   = 0;
rctLT.top    = 1111;
rctLT.right   = 115;
rctLT.bottom = 916;
var fRect = new ActiveXObject("OlvComWrapper.OlvFRectCom");
fRect.left = 100;
fRect.top = 900;
fRect.right = 300;
fRect.bottom = 700;
Cnfdnc = fzy.IsRectAboveOther( rctLT, fRect ); // Cnfdnc = 0.60
```

19.38 IsRectBelowOther

Is first rectangle's top is below other rectangle's bottom.

19.38.1 Prototype:

```
function IsRectBelowOther( Rect: OlvFRectCom, Other : OlvFRectCom)
:Confidence (defined at the top)
```

19.38.2 Parameters:

Rect: OlvFRectCom

Other: OlvFRectCom

19.38.3 Return value:

Confidence (defined at the top). It does not indicate the amount of overlap between the two. E.g. with 50% overlap the Confidence (return value) will probably be 0.0, while Confidence=0.50 will happen with almost 0% overlap.

19.38.4 Example:

```
fzy.DeltasRelative = false;

fzy.DeltaX = 60.0;

fzy.DeltaY = 80.0;

var fRect = new ActiveXObject("OlvComWrapper.OlvFRectCom");

fRect.left = 100;

fRect.top = 900;

fRect.right = 300;

fRect.bottom = 700;

var rctLT = new ActiveXObject("OlvComWrapper.OlvFRectCom");

rctLT.left   = 0;

rctLT.top    = 1111;

rctLT.right   = 115;

rctLT.bottom = 916;

Cnfdnc = fzy.IsRectBelowOther( fRect, rctLT); // Cnfdnc = 0.60

fzy.DeltaX = 0.3;

fzy.DeltaY = 0.4;

fzy.DeltasRelative = true;

var rctRB = new ActiveXObject("OlvComWrapper.OlvFRectCom");

rctRB.left   = 320;

rctRB.top    = 760;

rctRB.right   = 444;

rctRB.bottom = 666;

Cnfdnc = fzy.IsRectBelowOther( rctRB, fRect ); // Cnfdnc = 0.125
```

19.39 IsRectEncloseOther

Is first rectangle totally containing the other rectangle.

19.39.1 Prototype:

```
function IsRectEncloseOther( Rect: OlvFRectCom, Other : OlvFRectCom)
:Confidence (defined at the top)
```

19.39.2 Parameters:

Rect: OlvFRectCom

Other: OlvFRectCom

19.39.3 Return value:

Confidence (defined at the top). It does not indicate the amount of overlap between the two. E.g. with 50% overlap the Confidence (return value) will probably be 0.0, while Confidence=0.50 will happen with almost 100% overlap.

19.39.4 Example:

```
fzy.DeltasRelative = false;

fzy.DeltaX = 60.0;

fzy.DeltaY = 80.0;

var fRect = new ActiveXObject("OlvComWrapper.OlvFRectCom");

fRect.left = 100;

fRect.top = 900;

fRect.right = 300;

fRect.bottom = 700;

var rctIn = new ActiveXObject("OlvComWrapper.OlvFRectCom");

rctIn.left   = 115;

rctIn.top    = 916;

rctIn.right   = 320;

rctIn.bottom = 760;

 Cnfdnc = fzy. IsRectEncloseOther ( rctIn , fRect ); // Cnfdnc =
0.125
```

19.40 IsRectInsideOther

Is first rectangle totally inside the other rectangle

19.40.1 Prototype:

```
function IsRectInsideOther( Rect: OlvFRectCom, Other : OlvFRectCom)
:Confidence (defined at the top)
```

19.40.2 Parameters:

Rect: OlvFRectCom

Other: OlvFRectCom

19.40.3 Return value:

Confidence (defined at the top). It does not indicate the amount of overlap between the two. E.g. with 50% overlap the Confidence (return value) will probably be 0.0, while Confidence=0.50 will happen with almost 100% overlap.

19.40.4 Example:

```
fzy.DeltasRelative = false;

fzy.DeltaX = 60.0;

fzy.DeltaY = 80.0;

var fRect = new ActiveXObject("OlvComWrapper.OlvFRectCom");

    fRect.left = 100;

fRect.top = 900;

fRect.right = 300;

fRect.bottom = 700;

var rctIn = new ActiveXObject("OlvComWrapper.OlvFRectCom");

rctIn.left   = 115;

rctIn.top    = 916;

rctIn.right   = 320;

rctIn.bottom = 760;

Cnfdnc = fzy.IsRectInsideOther ( fRect, rctIn ); // Cnfdnc = 0.125
```

19.41 IsRectOutsideOther

Is first rectangle totally outside the other rectangle

19.41.1 Prototype:

```
function IsRectOutsideOther( Rect: OlvFRectCom, Other : OlvFRectCom)
:Confidence (defined at the top)
```

19.41.2 Parameters:

Rect: OlvFRectCom

Other: OlvFRectCom

19.41.3 Return value:

Confidence (defined at the top). It does not indicate the amount of overlap between the two. E.g. with 50% overlap the Confidence (return value) will probably be 0.0, while Confidence=0.50 will happen with almost 0% overlap.

19.41.4 Example:

```
fzy.DeltasRelative = false;

fzy.DeltaX = 60.0;

fzy.DeltaY = 80.0;

var fRect = new ActiveXObject("OlvComWrapper.OlvFRectCom");

    fRect.left = 100;

fRect.top = 900;

fRect.right = 300;

fRect.bottom = 700;

var rctRT = new ActiveXObject("OlvComWrapper.OlvFRectCom");

rctRT.left   = 320;

rctRT.top    = 1111;

rctRT.right   = 444;

rctRT.bottom = 916;

Cnfdnc = fzy.IsRectOutsideOther ( rctRT, fRect ); // Cnfdnc =
0.666666
```

19.42 IsRectIntersectOther

Is first rectangle intersects other rectangle

19.42.1 Prototype:

```
function IsRectIntersectOther( Rect: OlvFRectCom, Other :
OlvFRectCom) :Confidence (defined at the top)
```

19.42.2 Parameters:

Rect: OlvFRectCom

Other: OlvFRectCom

19.42.3 Return value:

Confidence (defined at the top). It does not indicate the amount of overlap between the two. E.g. with 50% overlap the Confidence (return value) will probably be 1.0, while Confidence=0.50 will happen with almost 0% overlap.

19.42.4 Example:

```
fzy.DeltaX = 0.3;

fzy.DeltaY = 0.4;

fzy.DeltasRelative = true;

var rctLT = new ActiveXObject("OlvComWrapper.OlvFRectCom");

rctLT.left   = 0;

rctLT.top    = 1111;

rctLT.right   = 115;

rctLT.bottom = 916;

var fRect = new ActiveXObject("OlvComWrapper.OlvFRectCom");

fRect.left = 100;

fRect.top = 900;

fRect.right = 300;

fRect.bottom = 700;

 Cnfdnc = fzy.IsRectIntersectOther (rctLT, fRect ); // Cnfdnc = 0.40

var rctRB = new ActiveXObject("OlvComWrapper.OlvFRectCom");

rctRB.left   = 320;

rctRB.top    = 760;
rctRB.right   = 444;

rctRB.bottom = 666;

Cnfdnc = fzy.IsRectIntersectOther( rctRB, fRect ); // Cnfdnc =
0.33333
```

19.43 IsRectRightUpperToOther

Is first rectangle's left border located Right to Other rectangle's right border, and first rectangle's bottom is above other rectangle's top.

19.43.1 Prototype:

```
function IsRectRightUpperToOther( Rect: OlvFRectCom, Other :
OlvFRectCom) :Confidence (defined at the top)
```

19.43.2 Parameters:

Rect: OlvFRectCom

Other: OlvFRectCom

19.43.3 Return value:

Confidence (defined at the top). It does not indicate the amount of overlap between the two. E.g. with 50% overlap the Confidence (return value) will probably be 0.0, while Confidence=0.50 will happen with almost 0% overlap.

19.43.4 Example:

```
fzy.DeltaX = 0.3;

fzy.DeltaY = 0.4;

fzy.DeltasRelative = true;

var rctRT = new ActiveXObject("OlvComWrapper.OlvFRectCom");

rctRT.left   = 320;

rctRT.top    = 1111;

rctRT.right   = 444;

rctRT.bottom = 916;

var fRect = new ActiveXObject("OlvComWrapper.OlvFRectCom");

fRect.left = 100;

fRect.top = 900;

fRect.right = 300;
fRect.bottom = 700;

Cnfdnc = fzy. IsRectRightUpperToOther (rctRT, fRect ); // Cnfdnc =
0.60
```

19.44 IsRectRightCenterToOther

Is first rectangle's left border located Right to Other rectangle's right border, and first rectangle's top and bottom are located between other rectangle's top and bottom.

19.44.1 Prototype:

```
function IsRectRightCenterToOther( Rect: OlvFRectCom, Other :
OlvFRectCom) :Confidence (defined at the top)
```

19.44.2 Parameters:

Rect: OlvFRectCom

Other: OlvFRectCom

19.44.3 Return value:

Confidence (defined at the top). It does not indicate the amount of overlap between the two. E.g. with 50% overlap the Confidence (return value) will probably be 0.0, while Confidence=0.50 will happen with almost 0% overlap.

19.44.4 Example:

```
fzy.DeltaX = 0.3;

fzy.DeltaY = 0.4;

fzy.DeltasRelative = true;

var rctRT = new ActiveXObject("OlvComWrapper.OlvFRectCom");

rctRT.left   = 320;

rctRT.top    = 1111;

rctRT.right   = 444;

rctRT.bottom = 916;

var fRect = new ActiveXObject("OlvComWrapper.OlvFRectCom");

fRect.left = 100;

fRect.top = 900;

fRect.right = 300;
fRect.bottom = 700;

Cnfdnc = fzy.IsRectRightCenterToOther(rctRT, fRect ); // Cnfdnc = 0.0

var rctRB = new ActiveXObject("OlvComWrapper.OlvFRectCom");

rctRB.left   = 320;

rctRB.top    = 760;

rctRB.right   = 444;

rctRB.bottom = 666;

Cnfdnc = fzy.IsRectRightCenterToOther(rctRB, fRect ); // Cnfdnc =
0.2875
```

19.45 IsRectRightLowerToOther

Is first rectangle's left border located Right to Other rectangle's right border, and first rectangle's top is below other rectangle's bottom.

19.45.1 Prototype:

```
function IsRectRightLowerToOther( Rect: OlvFRectCom, Other :
OlvFRectCom) :Confidence (defined at the top)
```

19.45.2 Parameters:

Rect: OlvFRectCom

Other: OlvFRectCom

19.45.3 Return value:

Confidence (defined at the top). It does not indicate the amount of overlap between the two. E.g. with 50% overlap the Confidence (return value) will probably be 0.0, while Confidence=0.50 will happen with almost 0% overlap.

19.45.4 Example:

```
fzy.DeltaX = 0.3;

fzy.DeltaY = 0.4;

fzy.DeltasRelative = true;

var rctRB = new ActiveXObject("OlvComWrapper.OlvFRectCom");

rctRB.left   = 320;

rctRB.top    = 760;

rctRB.right   = 444;
rctRB.bottom = 666;

var fRect = new ActiveXObject("OlvComWrapper.OlvFRectCom");

fRect.left = 100;

fRect.top = 900;

fRect.right = 300;

fRect.bottom = 700;

Cnfdnc = fzy. IsRectRightLowerToOther (rctRB, fRect ); // Cnfdnc =
0.125
```

19.46 IsRectCenterUpperToOther

Are first rectangle’s left and right borders located between other rectangle’s left and right borders, and first rectangle’s bottom is above other rectangle’s top.

19.46.1 Prototype:

function IsRectCenterUpperToOther(Rect: OlvFRectCom, Other : OlvFRectCom) :Confidence (defined at the top)

19.46.2 Parameters:

Rect: OlvFRectCom Other: OlvFRectCom

19.46.3 Return value:

Confidence (defined at the top). It does not indicate the amount of overlap between the two. E.g. with 50% overlap the Confidence (return value) will probably be 0.0, while Confidence=0.50 will happen with almost 0% overlap.

19.46.4 Example:

```
fzy.DeltaX = 0.3;

fzy.DeltaY = 0.4;

fzy.DeltasRelative = true;

var rctRT = new ActiveXObject("OlvComWrapper.OlvFRectCom");

rctRT.left   = 320;

rctRT.top    = 1111;

rctRT.right   = 444;

rctRT.bottom = 916;
```

```
var fRect = new ActiveXObject("OlvComWrapper.OlvFRectCom");
fRect.left = 100;
fRect.top = 900;
fRect.right = 300;
fRect.bottom = 700;
Cnfdnc = fzy. IsRectCenterUpperToOther (rctRT, fRect ); // Cnfdnc = 0.00
var rctCT = new ActiveXObject("OlvComWrapper.OlvFRectCom");
rctCT.left   = 115;
rctCT.top    = 1111;
rctCT.right  = 320;
rctCT.bottom = 916;
Cnfdnc = fzy. IsRectCenterUpperToOther (rctCT, fRect ); // Cnfdnc = 0.333333
```

19.47 IsRectCenterLowerToOther

Are first rectangle’s left and right borders located between other rectangle’s left and right borders, and first rectangle’s top is below other rectangle’s bottom.

19.47.1 Prototype:

```
function IsRectCenterLowerToOther( Rect: OlvFRectCom, Other :
OlvFRectCom) :Confidence (defined at the top)
```

19.47.2 Parameters:

Rect: OlvFRectCom

Other: OlvFRectCom

19.47.3 Return value:

Confidence (defined at the top). It does not indicate the amount of overlap between the two. E.g. with 50% overlap the Confidence (return value) will probably be 0.0, while Confidence=0.50 will happen with almost 0% overlap.

19.47.4 Example:

```
fzy.DeltaX = 0.3;
fzy.DeltaY = 0.4;
fzy.DeltasRelative = true;
```

```
var rctCB = new ActiveXObject("OlvComWrapper.OlvFRectCom");
rctCB.left   = 115;
rctCB.top    = 760;
rctCB.right   = 320;
rctCB.bottom = 666;
var fRect = new ActiveXObject("OlvComWrapper.OlvFRectCom");
  fRect.left = 100;
fRect.top = 900;
fRect.right = 300;
fRect.bottom = 700;
Cnfdnc = fzy.IsRectCenterLowerToOther( rctCB, fRect ); // Cnfdnc = 0.125
```

19.48 IsRectLeftUpperToOther

Is first rectangle’s right border located Left to Other rectangle’s left border, and first rectangle’s bottom is above other rectangle’s top.

19.48.1 Prototype:

```
function IsRectLeftUpperToOther( Rect: OlvFRectCom, Other : OlvFRectCom) :Confidence (defined at the top)
```

19.48.2 Parameters:

Rect: OlvFRectCom

Other: OlvFRectCom

19.48.3 Return value:

Confidence (defined at the top). It does not indicate the amount of overlap between the two. E.g. with 50% overlap the Confidence (return value) will probably be 0.0, while Confidence=0.50 will happen with almost 0% overlap.

19.48.4 Example:

```
fzy.DeltaX = 0.3;
fzy.DeltaY = 0.4;
fzy.DeltasRelative = true;
var rctLT = new ActiveXObject("OlvComWrapper.OlvFRectCom");
```

```
rctLT.left   = 0;
rctLT.top    = 1111;
rctLT.right   = 115;
rctLT.bottom = 916;
var fRect = new ActiveXObject("OlvComWrapper.OlvFRectCom");
    fRect.left = 100;
fRect.top = 900;
fRect.right = 300;
fRect.bottom = 700;
Cnfdnc = fzy. IsRectLeftUpperToOther (rctLT, fRect ); // Cnfdnc =
0.375
```

19.49 IsRectLeftCenterToOther

Is first rectangle’s right border located Left to Other rectangle’s left border, and first rectangle’s top and bottom are located between other rectangle’s top and bottom.

19.49.1 Prototype:

function IsRectLeftCenterToOther(Rect: OlvFRectCom, Other : OlvFRectCom) :Confidence (defined at the top)

19.49.2 Parameters:

Rect - OlvFRectCom

Other - OlvFRectCom

19.49.3 Return value:

Confidence (defined at the top). It does not indicate the amount of overlap between the two. E.g. with 50% overlap the Confidence (return value) will probably be 0.0, while Confidence=0.50 will happen with almost 0% overlap.

19.49.4 Example:

```
fzy.DeltaY = 0.4;
fzy.DeltasRelative = true;
var rctLC = new ActiveXObject("OlvComWrapper.OlvFRectCom");
rctLC.left   = 0;
rctLC.top    = 916;

rctLC.right   = 115;

rctLC.bottom = 760;

var fRect = new ActiveXObject("OlvComWrapper.OlvFRectCom");

    fRect.left = 100;

fRect.top = 900;

fRect.right = 300;

fRect.bottom = 700;

Cnfdnc = fzy. IsRectLeftCenterToOther ( rctLC, fRect ); // Cnfdnc =
0.375
```

19.50 IsRectLeftLowerToOther

Is first rectangle's right border located Left to Other rectangle's left border, and first rectangle's top is below other rectangle's bottom.

19.50.1 Prototype:

```
function IsRectLeftLowerToOther( Rect: OlvFRectCom, Other :
OlvFRectCom) :Confidence (defined at the top)
```

19.50.2 Parameters:

Rect - OlvFRectCom

Other - OlvFRectCom

19.50.3 Return value:

Confidence (defined at the top). It does not indicate the amount of overlap between the two. E.g. with 50% overlap the Confidence (return value) will probably be 0.0, while Confidence=0.50 will happen with almost 0% overlap.

19.50.4 Example:

```
fzy.DeltaY = 0.4;

fzy.DeltasRelative = true;

var rctLB = new ActiveXObject("OlvComWrapper.OlvFRectCom");

rctLB.left   = 0;

rctLB.top    = 760;

rctLB.right   = 115;
rctLB.bottom = 666;

var fRect = new ActiveXObject("OlvComWrapper.OlvFRectCom");

    fRect.left = 100;

fRect.top = 900;

fRect.right = 300;

fRect.bottom = 700;

Cnfdnc = fzy. IsRectLeftLowerToOther (rctLB, fRect ); // Cnfdnc =
0.125
```

20. OlvFuzzyRegExpCom

Regular Expression operations in consideration of *edit distance* (insertion, deletion, replacement).

20.1.1 Derived from:

20.1.2 Creation:

```
var regExpFuzzy = new
ActiveXObject("OlvComWrapper.OlvFuzzyRegExpCom");
```

21. PROPERTIES AND METHODS

21.1.1 Properties Read/Write:

n/a

21.1.2 Properties Read only:

All properties available only after successful execution of methods Match() or Grep()

21.2 MatchText : String

Actually matched Text

21.3 MatchBegin : String

Starting string of matched text

21.4 MatchEnd : String

Ending string of matched text

21.5 MatchDistortion : integer

“Edit distance”/distortion of matched Text (number of real errors)

21.6 MatchBeginOffset : integer

Offset of beginning of matched text in Text

21.7 MatchEndOffset : integer

Offset of end matched text in Text

21.7.1 Example:

See usage example below

21.7.2 Methods:

21.8 Match

21.8.1 Description:

Compares string with regular expression in consideration of “edit distance”.

21.8.2 Prototype:

```
function Match ( RegExpression : String
, Text : String
, TextLen : integer
, maxErrors : integer
, ProtectExpression : boolean )
: boolean
```

21.8.3 Parameters:

RegExpression - String, Expression/Text of source

Text - String, Text of target

TextLen - Integer, length of target text

maxErrors - Integer, max allowed "edit distance" (insertion, deletion, replacement)

ProtectExpression - Boolean, should RegExpression be treated as regular text (no control characters used)

21.8.4 Return value:

True if the Expression/Text matches the target, False otherwise

21.8.5 Example:

See usage example below

21.9 Grep

Searches for a substring the matches the regular expression when considering "edit distance".

21.9.1 Prototype:

```
function Grep ( RegExpression : String

, Text : String

, TextLen : integer

, maxErrors : integer

, ProtectExpression : boolean )

: boolean
```

21.9.2 Parameters:

RegExpression - String, Expression/Text of source

Text - String, Text of target

TextLen - Integer, length of target text

maxErrors - Integer, max allowed "edit distance" (insertion, deletion, replacement)

ProtectExpression - Boolean, should RegExpression be treated as regular text (no control characters used)

21.9.3 Return value:

True if the Expression/Text found in the target, False otherwise

21.9.4 Example:

see usage example below

21.9.5 NOTE:

If above functions fail, value of properties is inconsistent & unpredictable.

21.9.6 Usage example:

```
var regExpFuzzy = new
ActiveXObject("OlvComWrapper.OlvFuzzyRegExpCom");

function ExecuteFuzzyRegExpOnBlock( Text, UseGrep, RegExpAsString,
Threshold )

{

var TextLen = Text.length;

   var found = ( UseGrep )

? regExpFuzzy.Grep(sRegExp, Text, TextLen, Threshold, RegExpAsString)

: regExpFuzzy.Match(sRegExp, Text, TextLen, Threshold,
RegExpAsString);

if(found)

{

var Result = regExpFuzzy.MatchText;

OlvProgressIndicator.ReportLogText("\t"

   + regExpFuzzy.MatchBeginOffset + "+" + Result + "+"

+ regExpFuzzy.MatchEnd.length + " error="

   + regExpFuzzy.MatchDistortion);

if ( bRecurseOnBlock )

ExecuteFuzzyRegExpOnBlock( regExpFuzzy.MatchEnd, UseGrep,
RegExpAsString, Threshold );

}

return found;

}
```

22. OlvProgressIndicator

22.1.1 Description:

OlvProgressIndicator is a global object that can be used by script for progress reporting and/or logging.

22.1.2 Derived from:

22.1.3 Creation:

22.1.4 Comment:

Properties TotalPages and PageNumber look as non relevant for script.

Method BeginStage has three parameters. The last parameter (bToDisplay=true/false)   is definitly not relevant for script and will be removed. The second – number of sub-stages does not work correct if it is >=2. May be it should be removed too. Look Test_ProgressIndicator.js

23. PROPERTIES AND METHODS

23.1.1 Properties Read/Write:

23.2 PageNumber

23.2.1 Example:

23.2.2 Properties Read only:

23.3 TotalPages

23.3.1 Example:

23.3.2 Methods:

23.4 BeginStage

23.4.1 Description:

This method initializes and begins Progress Indicator

23.4.2 Prototype:

```
BeginStage ( stageName: String, SubstagesCount: Number)
```

23.4.3 Parameters:

stageName:

Unique string for stage identification.

SubstagesCount:

Number of substages. Values 0 and 1 are equalent and mean that no substages exist.

23.4.4 Return value:

23.4.5 Example:

See StageProgress

23.5 EndStage

This method stops Progress Indicator

23.5.1 Prototype:

```
EndStage ( stageName: String)
```

23.5.2 Parameters:

stageName:

Unique string for stage identification. Should be the same as in BeginStage method.

23.5.3 Return value:

23.5.4 Example:

See StageProgress

23.6 StageProgress

23.6.1 Description:

This method inform Progress Indicator what part of stage has been executed and causes refresh of progressing display

23.6.2 Prototype:

```
StageProgress ( percent: Number)
```

23.6.3 Parameters:

Percent:

Integer value from 0 to 100

23.6.4 Return value:

23.6.5 Example:

```
OlvProgressIndicator.BeginStage( "LongLoop", 0, true );

OlvProgressIndicator.StageStatus("First half of process");

Var bCancel = false;

for(i=0; i<20; i++)

{

for(j=0; j<1000000; j++)

{

         x++;

x = 0;

}

If(i == 10)

OlvProgressIndicator.StageStatus("First half of process");

OlvProgressIndicator.StageProgress(i*5);

bCancel = OlvProgressIndicator.IsCancelled();

if(bCancel)

break;

}

  If(bCancel)

OlvProgressIndicator.ReportWarning("Is cancelled");
```

```
else

{

var bActive = OlvProgressIndicator.IsActive(); // = TRUE

OlvProgressIndicator.EndStage( " LongLoop" );

bActive = OlvProgressIndicator.IsActive(); // = FALSE

if( bActive)

OlvProgressIndicator.ReportError("ProgressIndicator is active after
being cancelled");

}
```

23.7 StageStatus

23.7.1 Description:

This method informs Progress Indicator about process status and causes refresh of status display.

23.7.2 Prototype:

```
StageStatus ( statusName: String)
```

23.7.3 Parameters:

statusName:

String for displaying in Progress dialog as status of stage.

23.7.4 Return value:

23.7.5 Example:

See StageProgress

23.8 IsActive

23.8.1 Description:

Check if BeginStage() was called and EndStage() was not called.

23.8.2 Prototype:

```
IsActive() : Boolean
```

23.8.3 Parameters:

23.8.4 Return value:

TRUE if BeginStage() was called and EndStage() was not called; otherwise FALSE

23.8.5 Example:

See StageProgress

23.9 IsCancelled

23.9.1 Description:

Check if the script has been cancelled via "Cancel" button of Progress dialog.

23.9.2 Prototype:

```
IsCancelled() : Boolean
```

23.9.3 Parameters:

23.9.4 Return value:

TRUE if "Cancel" button was pressed and confirmed; otherwise FALSE.

Note that BeginStage, EndStage, StageStatus and StageProgress should not be called after cancelling.

23.9.5 Example:

See StageProgress

23.10 ReportLogText, ReportWarning and ReportError

23.10.1 Description:

Write message text to Log. Each message is is written together with its type depending on method. Three types are implemented: NOTIFY, WARNING and ERROR

23.10.2 Prototype:

```
ReportLogText( MessageText: String)

ReportWarning( MessageText: String)

ReportError( MessageText: String)
```

23.10.3 Parameters:

MessageText	Text that is written to Log

23.10.4 Return value:

23.10.5 Example:

See StageProgress

24. OlvRangeCom

Interval of two float numbers.

24.1.1 Derived from:

24.1.2 Creation:

```
var range = new ActiveXObject("OlvComWrapper.OlvRangeCom");
```

25. PROPERTIES AND METHODS

25.1.1 Properties Read/Write:

25.2 from:FLOAT

25.3 to:FLOAT

25.3.1 Properties Read only:

25.3.2 Methods:

25.4 IsInRange: Boolean

25.5 IsInFRange : Boolean

25.6 IsInTRange : Boolean

25.7 IsInFTRange : Boolean

Checks if a specified number is inside of range.

25.7.1 Prototype:

```
IsInRange(floatNumber: Float) : Boolean

IsInFRange(floatNumber: Float) : Boolean

IsInTRange(floatNumber: Float) : Boolean

IsInFTRange(floatNumber: Float) : Boolean
```

25.7.2 Parameters:

floatNumber

- a float number

25.7.3 Return value:

IsInRange - True if floatNumber is > range.from and < range.to;

IsInFRange - True if floatNumber is >= range.from and < range.to;

IsInTRange - True if floatNumber is > range.from and <= range.to;

IsInFTRange - True if floatNumber is >= range.from and <= range.to;

False otherwise

25.7.4 Usage example:

```
var FRange = new ActiveXObject("OlvComWrapper.OlvRangeCom");

var rFrom = 77.7;

var rTo   = 99.9;

FRange.from  = rFrom;

FRange.to    = rTo;

Range_0.from = FRange.from;
```

```
Range_0.to   = FRange.to;

var tmp0 = FRange.IsInRange( rFrom ); // = false

var tmp1 = FRange.IsInRange( rTo );   // = false

var tmp0 = FRange.IsInFRange( rFrom ); // = true

var tmp1 = FRange.IsInFRange( rTo );   // = false

var tmp0 = FRange.IsInTRange( rFrom ); // false

var tmp1 = FRange.IsInTRange( rTo );   // true

var tmp0 = FRange.IsInFTRange( rFrom ); // true

var tmp1 = FRange.IsInFTRange( rTo ); // true
```

26. OlvRectCom

26.1.1 Description:

OlvFRectCom object defines the top-left and bottom-right points of a rectangle.

When specifying a **OlvFRectCom** object, you must be careful to construct it so that it is normalized — in other words, such that the value of the left coordinate is less than the right and the bottom is less than the top.

26.1.2 Derived from:

26.1.3 Creation:

var rect = new ActiveXObject("OlvComWrapper.OlvRectCom");

27. PROPERTIES AND METHODS

27.1.1 Properties Read/Write:

27.2 left

Specifies the x-coordinate of a left side of rectangle

27.3 right

Specifies the x-coordinate of a right side of rectangle

27.4 bottom

Specifies the y-coordinate of a bottom side of rectangle

27.5 top

Specifies the y-coordinate of a top side of rectangle

27.5.1 Example:

```
// Create rectangle
var Rect = new ActiveXObject("OlvComWrapper.OlvFRectCom");
// Set values of coordinates
Rect.left = 10;
Rect.top = 10.5;
Rect.right = 55.5;
Rect.bottom = 5.5;
// Get values
var xLeft = Rect.x;  // xLeft = 10
var xRight = Rect.right; // xRight = 55.5
var yTop = Rect.top; // yTop = 10.5
var yBottom = Rect.bottom; // yBottom = 5.5
```

27.5.2 Properties Read only:

27.5.3 Methods:

27.6 Width

27.6.1 Description:

Computes width of rectangle.

27.6.2 Prototype:

```
Width(): Number
```

27.6.3 Parameters:

27.6.4 Return value:

Width of rectangle

27.6.5 Example:

See Square

27.7 Height

27.7.1 Description:

Computes height of rectangle.

27.7.2 Prototype:

```
Height(): Number
```

27.7.3 Parameters:

27.7.4 Return value:

Height of rectangle

27.7.5 Example:

See Square

27.8 Square

27.8.1 Description:

Computes square of rectangle.

27.8.2 Prototype:

Square(): Number

27.8.3 Parameters:

27.8.4 Return value:

Square of rectangle

27.8.5 Example:

```
// Create Rectangle

var Rect = new ActiveXObject("OlvComWrapper.OlvFRectCom");

// Set values of coordinates

Rect.left = 10;

Rect.top = 10.5;

Rect.right = 55.5;

Rect.bottom = 5.5;

var w = Rect.Width();  // w = 45.5

var h = Rect.Height();  // h = 5

var s = Rect.Square();  // s = 235
```

27.9 IsPointInside

27.9.1 Description:

Determines whether the specified point lies within rectangle.

27.9.2 Prototype:

```
IsPointInside(point: OlvFPointCom): Boolean
```

27.9.3 Parameters:

Point - Contains a OlvFPointCom object

27.9.4 Return value:

True if the point lies within rectangle otherwise false.

If point lies on one of the sides it considered to be inside.

27.9.5 Example:

```
// Set point

point  .x = 300.9;

point  .y = 341.9;

// Set Rectangle

               Rect.left = 0;
Rect.bottom = 134;

Rect.right = 301;

Rect.top = 342;

var b1 = Rect.IsPointInside(point  ); // b1 = true
```

27.10 IsRectInside

27.10.1 Description:

Determines whether the specified rectangle lies within a source rectangle.

27.10.2 Prototype:

```
IsRectInside (otherRect: OlvFRectCom): Boolean
```

27.10.3 Parameters:

otherRect - Contains a OlvFRectCom object

27.10.4 Return value:

True if <otherRect> lies within rectangle otherwise false.

27.10.5 Example:

```
// Set Rectangles

Rect.left = 0;

Rect.bottom = 134;

Rect.right = 301;

Rect.top = 342;

otherRect.left = 10;

Rect.bottom = 200;

Rect.right = 20;

Rect.top = 300;

var b = Rect.IsRectInside(otherRect  ); // b = true
```

27.11 IsIntersect

27.11.1 Description:

Determines whether the specified rectangle lies within a source rectangle.

27.11.2 Prototype:

```
IsIntersect(otherRect: OlvFRectCom): Boolean
```

27.11.3 Parameters:

otherRect - Contains a OlvFRectCom object

27.11.4 Return value:

True if <otherRect> intersects with a source rectangle otherwise false.

27.11.5 Example:

See UniteRect

27.12 IntersectRect

27.12.1 Description:

Makes the dimensions of source rectangale equal to its intersection with the specified rectangle.

27.12.2 Prototype:

```
Rect.IntersectRect(otherRect: OlvFRectCom ): Boolen
```

27.12.3 Parameters:

otherRect - Contains a OlvFRectCom object

27.12.4 Return value:

True if <otherRect> intersects with a source rectangle otherwise false.

27.12.5 Example:

See UniteRect

27.13 Intersect2Rects

27.13.1 Description:

Makes the dimensions of source rectangale equal to intersection beween two specified rectangles.

27.13.2 Prototype:

```
Intersect2Rects(rectA: OlvFRectCom, rectB: OlvFRectCom): Boolean
```

27.13.3 Parameters:

rectA - Contains a OlvFRectCom object

rectB - Contains a OlvFRectCom object

27.13.4 Return value:

True if rectA and rectB intersect otherwise false.

27.13.5 Example:

See UniteRect

27.14 UniteRect

27.14.1 Description:

Makes the dimensions of source rectangle equal to its union with the specified rectangles

27.14.2 Prototype:

```
UniteRect(otherRect: OlvFRectCom)
```

27.14.3 Parameters:

otherRect - Contains a OlvFRectCom object

27.14.4 Return value:

27.14.5 Example:

```
// Set Rectangles
Rect.left = 0;
Rect.bottom = 134;
Rect.right = 301;
Rect.top = 200;
otherRect.left = 10;
otherRect.bottom = 200;
otherRect.right = 20;
otherRect.top = 300;
RectA.left = 0;
RectA.bottom = 134;
RectA.right = 301;
   RectA.top = 200;
RectB.left = 20;
RectB.bottom = 100;
RectB.right = 500;
RectB.top = 150;
// Set to intersection with other rectangle
var b1 = Rect.IntersectRect(otherRect  ); // b1 = true
// Results
var left = Rect.left; // left = 10
var top = Rect.top; // top = 200
var right = Rect.right; // right = 20
var bottom = Rect.bottom; // bottom = 200
// Set to intersection of two other rectangles
var  b2 = Rect.Intersect2Rects(rectA, otherRect); // b2 = true
// Results
var left = Rect.left; // left = 10
   var top = Rect.top; // top = 200
var right = Rect.right; // right = 20
var bottom = Rect.bottom; // bottom = 200
```

```
// Set to union with other rectangle

RectB.UniteRect(RectA);

// Results

var left = RectB.left; // left = 0

var top = RectB.top; // top = 200

var right = RectB.right; // right = 500

var bottom = RectB.bottom; // bottom = 100
```

27.15 UnitePoint

27.15.1 Description:

Makes the dimensions of source rectangle equal to its union with the specified point

27.15.2 Prototype:

```
UnitePoint(point: OlvFPointCom)
```

27.15.3 Parameters:

Point - Contains a OlvFPointCom object

27.15.4 Return value:

27.15.5 Example:

```
// Set rectangle

RectB.left = 20;

RectB.bottom = 100;

RectB.right = 500;

RectB.top = 150;

// Set point

Point.x = 0;

Point.y = 120;

// Unite

RectB.UnitePoint(point);

// Results
```

```
var left = RectB.left; // left = 0

var top = RectB.top; // top = 150

var right = RectB.right; // right = 500

var bottom = RectB.bottom; // bottom = 100
```

27.16 OffsetRect

27.16.1 Description:

Moves rectangle by the specified offsets

27.16.2 Prototype:

```
OffsetRect(x: Number, y: Number)
```

27.16.3 Parameters:

X - Specifies the amount to move left or right. It must be negative to move left.

Y - Specifies the amount to move up or down. It must be negative to move down.

27.16.4 Return value:

27.16.5 Example:

```
// Set rectangle

RectB.left = 20;

RectB.bottom = 100;

RectB.right = 500;

RectB.top = 150;

// Set offsets

Var x = -20;

Var y = 100;

// Move

RectB. OffsetRect (point);

// Results

var left = RectB.left; // left = 0
var top = RectB.top; // top = 250

var right = RectB.right; // right = 480

var bottom = RectB.bottom; // bottom = 200
```

27.17 InflateRect

27.17.1 Description:

inflates rectangle by moving its sides away from its center.

27.17.2 Prototype:

```
InflateRect(x: Number, y: Number)
```

27.17.3 Parameters:

X - Specifies the amount to inflate the left and right sides of rectangle

Y - Specifies the amount to inflate top and bottom of rectangle

27.17.4 Return value:

27.17.5 Example:

See IsEqual

27.18 IsRectSimilarTo

27.18.1 Description:

Determines whether the specified rectangle is approximately equal to the source rectangle by comparing the coordinates of their upper-left and lower-right corners

27.18.2 Prototype:

```
IsRectSimilarTo(otherRect: OlvFRectCom, Eps: Number): Boolean
```

27.18.3 Parameters:

otherRect

Contains a OlvFRectCom object

eps

float value

27.18.4 Return value:

True if values of left, top, right and bottom of the source rectangle and of <otherRect> differ not more than by <Eps> otherwise false.

27.18.5 Example:

See IsEqual

27.19 IsEqual

Determines whether source rectangle is equal to the specified rectangle by comparing the coordinates of their upper-left and lower-right corners

27.19.1 Prototype:

```
IsEqual(otherRect: OlvFRectCom): Boolean
```

27.19.2 Parameters:

otherRect - Contains a OlvFRectCom object

27.19.3 Return value:

True if values of left, top, right and bottom of the source rectangle and <otherRect> are equal otherwise false.

27.19.4 Example:

```
// Set source rectangle

Rect.left = 20;

Rect.bottom = 100;

Rect.right = 500;

Rect.top = 150;

// Set other rectangle

RectB.left = 20;

RectB.bottom = 100;

RectB.right = 500;

RectB.top = 150;

// Are they equal?
```

```
Var b = Rect.IsEqual(Rect); // b = true

// Inflate by  -5, 20

RectB. InflateRect(-5, 20);

// Results

var left = RectB.left; // left = 25

var top = RectB.top; // top = 170

var right = RectB.right; // right = 495

var bottom = RectB.bottom; // bottom = 80

// Are rectangles approximately the same?

Var b1 = Rect. IsRectSimilarTo(RectB, 10); // b1 = false

Var b2 = Rect. IsRectSimilarTo(RectB, 20); // b1 = true
```

28. OlvTextStyleInfo

28.1.1 Description:

This object that describes a text style and counts how many times it appears.

28.1.2 Derived from:

28.1.3 Creation:

You can get a meaningful OlvTextStyleInfo **only** as an output from some functions. It has no writable properties, so you **can not** build a meaningful object by yourself.

28.2 PROPERTIES AND METHODS

28.2.1 Properties Read/Write:

28.2.2 Properties Read only:

28.3 FontName : string

Name of the font

28.4 FontSize : float

Size of the font

28.5 FontSizeDeviation : float

The allowed deviation of the **measured** font size of some actual printed text from the current **FontSize** by **FontSizeDeviation/2**.

28.6 FontFlags : integer

A summation of numeric values that represent a combination of the following string values/properties: “scanned”, “bold”, “italic” and “colored”

28.7 NumOfPages : integer

Number of pages (OlvPageCom), from those that were checked, containing current text style.

28.8 NumOfTextBlocks : integer

Number of TextBlocks (OlvTextBlockCom), from those that were checked, containing current text style.

28.9 NumOfTextLines : integer

Number of TextLines (OlvTextLineCom), from those that were checked, containing current text style.

28.10 NumOfQuads : integer

Number of Quads (OlvQuadCom), from those that were checked, containing current text style.

28.10.1 Example:

28.10.2 Methods:

29. OlvLinkCom

This object represents a link to another page, another document, or a Web link (e-mail or URL).

The OlvLink object of MDLocation type represents the location of Metadata in the Document.

29.1.1 Derived from:

OlvPropertySet

OlvElementProvider

29.1.2 Creation:

Get an existing link or create a new one using the corresponding functions of OlvPageCom

29.2 PROPERTIES AND METHODS

29.2.1 Properties Read/Write:

29.3 LinkType: number

One of the following:

- OlvLinkTypePage
- OlvLinkTypeURL
- OlvLinkTypeMail
- OlvLinkTypeDocument
- OlvLinkTypeMDLocation

29.4 TargetPageNumber:number

0-based index of target page to be linked

29.5 TargetString:string

Relevant when linking to a document, URL or e-mail

29.6 TargetBox:OlvFRectCom

Optional to link page – box on target page

29.7 TargetZoomBox:OlvFRectCom

Optional to link page – box on target page to zoom to when linking

29.8 ZoomToPageWidth:boolean

Optional to link page – to zoom to page width when following the link

29.9 Properties Read only:

29.9.1 Methods:

29.9.2 Usage Examples:

```
var Page = OlvDoc.GetPage(0);

var newLink = Page.CreateLink(OlvLinkTypePage, rLinkBox);

newLink.TargetPageNumber = 1;
```

30. Appendix A: Script Files

This section includes the full content of the following ORL script files:

30.1 XMDTeplate.js

```
// XMDTemplate.js

//==============
// Global scope
//==============
debugger

var strGlobalMsg = "";
var strStage = "ScriptProcess";

try
{
   // 'ScriptProcess' can be replaced by any name for a stage
   eval( GetGlobalDefinitions() );
   StartScript();

   MainCode(); // ... To be filled with user's desired code ...

   EndScript();
}
catch( exception )
{
   if (exception.description == "OlvException: Failed to load include
file")
```

```
         OnIncludeFileException();
   else
         OnException( exception ); // Exception handling
}

function OnIncludeFileException(description)
{
   OlvProgressIndicator.BeginStage(strStage, 0, true);
   OlvProgressIndicator.ReportError(strGlobalMsg);
   OlvProgressIndicator.EndStage(strStage);
}

//============================================================
// To be filled with user's desired code
//============================================================
function MainCode()
{
   // ... TBD ...
}

//============================================================
// Get Global Definitions
//============================================================
function GetGlobalDefinitions( sStageName )
{
    var fso = new ActiveXObject("Scripting.FileSystemObject");
    var flagForReading = 1;
    var fileName =
"S:\\XMD\\Develop\\AutoProcess\\XMDScripts\\XMD4Include.js";
    if (!fso.FileExists(fileName))
    {
         strGlobalMsg = "Exception: Failed to load include file < " +
fileName + " >";
         throw new Error("OlvException: Failed to load include
file");
    }

   var IncludeFile = fso.OpenTextFile(fileName, 1/*ForReading*/,
true);
   return IncludeFile.ReadAll();
}
```

30.2 XMD4Include.js

```
// XMD4Include.js

/////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////
// Property Flags
varOlvFlag_Temporary  =    0x00000001; // The property is not saved
in PDF Dictionary
var OlvFlag_Attribute  =   0x00000002; // Value of Property is
'confidence' (from [0.1]
var OlvFlag_AboveEntityLevel =   0x00000008 // PropName is
BSPropName_TOC_HEADING_LEVEL and value is upper than Article Title

// Property MD Flags - means that this property is to be part of MD
of relevant level

varOlvFlag_MD4Doc    =           0x00000100; // MetaData for Document
varOlvFlag_MD4Page   =           0x00000200; // MetaData for Page
varOlvFlag_MD4Entity =           0x00000400; // MetaData for Entity
varOlvFlag_MD4Primitive =  0x00001000; // MetaData for Block
varOlvFlag_MD4Tag          =     0x00002000; // MetaData for Tag

// Property Entity/Element/Sub-element Flags
// NOTE! These flags request presence of OlvFlag_Attribute:
// and only one of these flags can be present in OlvProperty.
var OlvFlag_EntType        =     0x00100000; //Property name is
ENTITY_TAG_NAME(“article”, “picture” or “ad”).
var OlvFlag_EntSubType     =     0x00200000; //Property name is
SUB_ENTITY_TAG_NAME(“Table” etc.).
var OlvFlag_ElemType =     0x00400000; //Property name is
ELEMENT_TAG_NAME (“article.content” ,”article.hedLine.hl1” etc.)
// If no property with OlvFlag_ElemType is present - properties with
OlvFlag_ElemSubType are ignored
var OlvFlag_ElemSubType    =     0x00800000; //Property name is
SUB_ELEMENT_TAG_NAME (“Chapter”, “Paragraph” etc.)
/////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////

/////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////
// Property Names

// Value is a string.
// All OlvBlocks having this property with the same value and only
they are included to the same Entity.
// Different chunks (continuations) are to be created for case of
different pages.
var OlvPropName_ENTITY_ID_ALL =              "ENTITY_ID_ALL";
// Value is a string.
// All OlvBlocks having this property with the same value are
included to the same Entity.
// Different chunks (continuations) are created for case of different
pages.
var OlvPropName_ENTITY_ID_PART =       "ENTITY_ID_PART";
```

```
// Value is a string.
// It defines that target XMDEntity should be embedded to another one
with ENTITY_ID value equal to it.
var OlvPropName_PARENT_ENTITY_ID =            "PARENT_ENTITY_ID";
// Value is a string.
// It defines that target XMDEntity should be embedded to a primitive
with PRIMITIVE_ID value equal to it.
var OlvPropName_PARENT_PRIMITIVE_ID =  "PARENT_PRIMITIVE_ID";
// Value is integer.
// All XMDPrimitives of the same XMDEntity are supposed to be sorted
according this value.
var OlvPropName_PRIMITIVE_SEQ_NO =            "PRIMITIVE_SEQ_NO";
// Value is TRUE or FALSE. If TRUE then no XMDPrimitive is to be
created.
var OlvPropName_PRIMITIVE_IGNORE  =           "PRIMITIVE_IGNORE";
// Value is string.
// OlvBlocks with the same "PRIMITIVE_ID" are not supposed to be
united with others to the same XMDPrimitive.
var OlvPropName_PRIMITIVE_ID =                "PRIMITIVE_ID";
// Value is string that is one of PrimitiveTagIDs defined in XMD.ini
or Profile in [PrimitiveTag]
// For example: "primtAgency", "primtBodyTitle", "primtAuthorName" -
one
// The OlvTag should be used for creating XMDPrimitiveTag with Name =
value of property
var OlvPropName_PRIMTIVE_TAG_ID =      "PRIMITIVE_TAG_ID";
// Value does not matter.
// Presence of this property means that OlvTag should be extracted by
Final Segmentation to a separate XMDPrimitive
var OlvPropName_TAG_EXTRACT_AS_PRIMITIVE =
"TAG_EXTRACT_AS_PRIMITIVE";
// Value is integer. (OlvTag only)
// Used for creation Link to Page from OlvTag
var OlvPropName_TAG_LINK_PAGE_NO  =
   "TAG_LINK_PAGE_NO";
// Value is string (OlvTag only)
// Link to Page is to be created from box of OlvTag
var OlvPropName_TAG_LINK_PAGE_LABEL =         "TAG_LINK_PAGE_LABEL";
// Value is rectangle (OlvTag only with property
OlvPropName_TAG_LINK_PAGE_LABEL or OlvPropName_TAG_LINK_PAGE_NO)
// Target Box for "See Box" Link that is to be created from box of
OlvTag
var OlvPropName_TAG_LINK_PAGE_TARGET_BOX =    "TAG_LINK_PAGE_BOX";
// Value does not matter.
// Presence of this property means that BSBlock is part of Page
Footer.
var OlvPropName_PAGE_FOOTER = "PAGE_FOOTER";
// Value does not matter.
// Presence of this property means that BSBlock is part of Page
Header.
var OlvPropName_PAGE_HEADER = "PAGE_HEADER";
// Value does not matter.
// Presence of this property means that Page contain TOC area.
var OlvPropName_PAGE_TOC = "PAGE_TOC";
// Value does not matter.
// Presence of this property means that Block is part of Page
Header..
```

```
var OlvPropName_TOC_AREA = "TOC_AREA";
// Value is string (OlvPage only)
// Used to set Page Label
var OlvPropName_PAGE_LABEL = "PAGE_LABEL";
// Value is string. (OlvPage only)
// Used to set Page Section
var OlvPropName_PAGE_SECTION = "PAGE_SECTION";
// var OlvPropName_LINK_MAIL, LINK_WWW, LINK_OTHER_ISSUE ???
// Value is bool
// Is set by Final Segmentation with flag OlvFlag_Temporary and
deleted after processing page
var OlvPropName_USED_FOR_PRIMTIVE_TAG ="USED_FOR_PRIMITIVE_TAG";
// Value is Integer.
// Is set by TOC Recognition process and is not used any more
var OlvPropName_TOC_HEADING_LEVEL =    "TOC_HEADING_LEVEL";
// Value is a full path to Document
var OlvPropName_FILE_PATH ="FILE_PATH";
/////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////

// Block Types
var OlvBlockTypeUnknown = 0;
var OlvBlockTypeText  = 1;
var OlvBlockTypeGraph = 2;
var OlvBlockTypeVerSep      = 4;
var OlvBlockTypeHorSep      = 8;
var OlvBlockTypeTable = 16;  //0x10
var OlvBlockTypeAll         = 255; //0xff

// Text Orientation
var OlvOrientation_Unknown = 0;
var OlvOrientation_Portrait = 1;
var OlvOrientation_Landscape = 2;

// Blocks Sorting Modes
var OlvReadingOrder              = 1; // Sort by reading order of
blocks
var OlvPageNumTopOrder           = 2; // Sort by Page Number and then
by Top Order

// Following definitions are used in Filtering
// Page Status
var OlvUndefinedPageStatus = 0;
var OlvPageSingle = 1;
var OlvPageLeft = 2;
var OlvPageRight = 3;

// Order Relations
var OlvOrdUndefined = 0;
var OlvOrdBefore = 1;
var OlvOrdOverlap = 2;
var OlvOrdInside = 3;
var OlvOrdAfter = 4;

// Order Regime
var OlvUndefinedOrder = 0;
var OlvReadingOrder = 1;
var OlvPageAndTopOrder = 2;

// Location Relations (any OR combination)
var OlvLocUndefined  =     0x0000;

var OlvLocAbove      =     0x0001;
var OlvLocBelow      =     0x0002;
var OlvLocLeft       =     0x0004;
var OlvLocRight      =     0x0008;
var OlvLocHorCenter  =     0x0010;
var OlvLocVerCenter  =     0x0020;

var OlvLocInside     =     0x0040;
var OlvLocOverlap    =     0x0080;
var OlvLocOutside    =     (OlvLocAbove | OlvLocBelow | OlvLocLeft |
OlvLocRight); // Above or Below or Left or Right

var strStageNameGlobal = "Executing Script";

// For loggging
var arrMsgList = new Array();
var arrMsgListStatus = new Array();
var AllMethodsCount = 0;
var ErrorsCount = 0;

var NOTIFY = 0;
var OK = 1;
var FAIL = 2;
var EPSILON = 0.00000001;

//===================================================================
===========================
// Macrose for numbers
function Min( A, B )
{
   return ( A < B ) ? A : B;
}
function Max( A, B )
{
   return ( A > B ) ? A : B;
}
function AreEqual( A, B ) //for floats/doubles
{
   var diff = A - B;
   return ( -EPSILON < diff && diff < EPSILON );
}
function IsZero( A ) //for floats/doubles
{
   return      AreEqual( A, 0.0 )
}

//===================================================================
===========================
// Create Array Object from safeArray (created by COM API).
function toArray( safeArray )
{
         var newVBarray = new VBArray( safeArray );
         return newVBarray.toArray();
}
// Create safeArray (for passing to COM API)from Array Object.
function toSafeArray( array )
{
         var newDict = new ActiveXObject("Scripting.Dictionary");
         for( var i = 0; i < array.length; ++i )
               newDict.Add(i, array[i]);
         return newDict.Items();
}
//===================================================================
=========
// sStageName - name of stage
// nSubstages - number of substages
//===================================================================
=========
function StartScript(sStageName, nSubstages)
{
   strStageNameGlobal = sStageName
   OlvProgressIndicator.BeginStage( strStageNameGlobal, nSubstages,
true );
   OlvProgressIndicator.ReportLogText(" ");
   OlvProgressIndicator.ReportLogText(" ");
   OlvProgressIndicator.ReportLogText("******************************
*********************************************");
   OlvProgressIndicator.ReportLogText("******************************
*********************************************");
   OlvProgressIndicator.ReportLogText("******************************
*********************************************");
}

//===================================================================
===================
// ASSUMES: there exist a global 'strStagename' { the name (string)
of the Stage }
//             It is defined in this script as "Executing Script" but
can be redefined in calling script
//===================================================================
===================
function EndScript()
{
   OlvProgressIndicator.ReportLogText("******************************
*********************************************");
   OlvProgressIndicator.ReportLogText("******************************
*********************************************");
   OlvProgressIndicator.ReportLogText("******************************
*********************************************");
   OlvProgressIndicator.ReportLogText(" ");
   OlvProgressIndicator.ReportLogText(" ");
   OlvProgressIndicator.EndStage( strStageNameGlobal );
}

//===================================================================
=========
// Exception handling
// ASSUMES: there exist a global 'strStagename' { the name (string)
of the lStage }
```

```
//              It is defined by XMDTemplate.js
//====================================================================
=========
function OnException( exception )
{
   ComputeAndPrintResults("Undefined");
   var description = exception.description;
   OlvProgressIndicator.ReportError( "  ERROR exception: " +
description );
   EndScript();
}

function AddMessage(Type, Message, AppendValue)
{
   if (AppendValue != null)
         arrMsgList.push(Message + ". " + AppendValue);
   else
         arrMsgList.push(Message);

   arrMsgListStatus.push(Type);
}

function WriteMessagesToLog(ErrorsOnly)
{
   for (var i = 0; i < AllMethodsCount; ++i)
   {
         if (arrMsgListStatus[i] == NOTIFY)
         {
               if ((ErrorsOnly && (i + 1 < AllMethodsCount) &&
(arrMsgListStatus[i + 1] == FAIL)) ||
                     !ErrorsOnly)
               {

   OlvProgressIndicator.ReportLogText(arrMsgList[i]);
               }

               continue;
         }

         if (arrMsgListStatus[i] == OK)
         {
               if (!ErrorsOnly)
               {
                     OlvProgressIndicator.ReportLogText("OK. " +
arrMsgList[i]);
                     OlvProgressIndicator.ReportLogText(" ");
               }
         }
         else
         {
               OlvProgressIndicator.ReportWarning("ERROR. " +
arrMsgList[i]);
               OlvProgressIndicator.ReportLogText(" ");
         }
   }
}
```

```
function ComputeAndPrintResults(Num)
{
   if (Num == "Undefined")
         return;

   AllMethodsCount = arrMsgList.length;
   for (var i = 0; i < AllMethodsCount; ++i)
   {
         if (arrMsgListStatus[i] == FAIL)
               ++ErrorsCount;
   }

   OlvProgressIndicator.ReportLogText("ALL Methods/Properties tested:
" + Num);
   OlvProgressIndicator.ReportLogText("Errors: " + ErrorsCount);

   if (ErrorsCount)
   {
         OlvProgressIndicator.ReportLogText(" ");

   OlvProgressIndicator.ReportLogText("******************************
*********************************************");
         OlvProgressIndicator.ReportLogText("List of Errors: ");
         OlvProgressIndicator.ReportLogText(" ");

         WriteMessagesToLog(true); // Errors only


   OlvProgressIndicator.ReportLogText("******************************
*********************************************");
         OlvProgressIndicator.ReportLogText(" ");
   }

   if (OlvDoc.FullLog)
   {
         OlvProgressIndicator.ReportLogText(" ");
         OlvProgressIndicator.ReportLogText("List of ALL
Methods/Properties: ");
         OlvProgressIndicator.ReportLogText(" ");

         WriteMessagesToLog(false); // Full list


   OlvProgressIndicator.ReportLogText("******************************
*********************************************");
   }
}
```