



Bundesamt für Landestopographie  
Office fédéral de topographie  
Ufficio federale di topografia  
Federal office of topography

Kompetenzzentrum INTERLIS  
Centre de compétence INTERLIS  
Centro di competenza INTERLIS  
Center of competence INTERLIS

# INTERLIS Version 2.0

## Symbologiemodelle

## Benutzerhandbuch

Ausgabe 1.0 vom 2000-12-13 (deutsch)



Bundesamt für Landestopographie  
Eidg. Vermessungsdirektion, Kompetenzzentrum INTERLIS  
Seftigenstrasse 264, CH-3084 Wabern, Fax +41 31 963 22 97  
Web [www.swisstopo.ch](http://www.swisstopo.ch) oder [www.gis.ethz.ch](http://www.gis.ethz.ch), E-Mail [interlis@lt.admin.ch](mailto:interlis@lt.admin.ch)

*Copyright © 2000 Bundesamt für Landestopographie, CH-3084 Wabern*

Alle mit © bezeichneten Namen sind mit dem Copyright des jeweiligen Autors oder Herstellers geschützt. Vervielfältigungen sind *ausdrücklich erlaubt* solange der Inhalt unverändert bleibt.

## Inhalt

<b>1</b>	<b>Einleitung.....</b>	<b>3</b>
<b>2</b>	<b>Überblick Modelle, Daten, Grafik, Symbologie.....</b>	<b>5</b>
<b>3</b>	<b>Die Symbologiemodelle in der Übersicht.....</b>	<b>7</b>
3.1	Überblick.....	7
3.2	Die vordefinierte CLASS SIGN.....	7
3.3	Das Symbologiemodell BasicSymbology.....	7
3.4	Das Symbologiemodell ExtendedSymbology.....	8
<b>4</b>	<b>Die Symbologiemodelle im Detail .....</b>	<b>9</b>
4.1	Einleitung .....	9
4.2	Die vordefinierte CLASS SIGN.....	9
4.3	Das Symbologiemodell BasicSymbology.....	10
4.4	Das Symbologiemodell ExtendedSymbology .....	10
4.4.1	Einleitung .....	11
4.4.2	Geometrische Ausdehnungen/Ausrichtungen .....	11
4.4.3	Farb-Bibliothek.....	16
4.4.4	Polylinienattribut-Bibliothek.....	17
4.4.5	Font-Bibliothek .....	17
4.4.6	Linienarten-Bibliothek .....	21
4.4.7	Textsignatur .....	26
4.4.8	Symbolsignatur .....	29
4.4.9	Liniensignatur.....	31
4.4.10	Flächensignatur .....	33
<b>5</b>	<b>Anhänge.....</b>	<b>35</b>

# 1 Einleitung

Dieses Benutzerhandbuch ist als Ergänzung zum

INTERLIS Version 2.0 Referenzhandbuch [1]

zu betrachten. Im vorliegenden Dokument werden die in INTERLIS Version 2.0 vorhandenen Möglichkeiten zur Definition der grafischen Darstellungen von Objekten näher betrachtet und beschrieben. Der Leser dieses Dokumentes sollte die wichtigsten Elemente der Beschreibungssprache INTERLIS Version 2.0 kennen. Es wird hier nicht weiter auf die Syntax oder Bedeutung einzelner Sprachelemente von INTERLIS Version 2.0 eingegangen. Für entsprechende Fragen verweisen wir auf das oben erwähnte Referenzhandbuch.

Auf konzeptionellen Niveau werden räumliche Daten meist in einer mehr oder weniger formellen Modellierungssprache oder Modellierungsumgebung beschrieben. Diese Beschreibung beinhaltet die Definitionen der Sachdaten und der geometrischen Lage der räumlichen Daten. Räumliche Objekte müssen häufig in Bildschirm- und Plandarstellungen grafisch dargestellt werden. Die Definitionen dieser grafischen Darstellung findet in der Regel nicht auf dem konzeptionellen Niveau statt, sondern wird eher auf dem Niveau physisches Schema in komplexen Systemkonfigurationen implementiert, die meist proprietär und schwer verständlich sind.

Als wesentliche Neuerung von INTERLIS Version 2.0 lassen sich diese grafischen Darstellungen ebenfalls auf einem präzisen konzeptionellen Niveau beschreiben. Damit wird INTERLIS Version 2.0 der Tatsache gerecht, dass die grafische Darstellung ebenfalls ein Bestandteil der Objektstrukturen dieser Daten sind. So wird zum Beispiel ein Schacht nicht nur durch seine Sachdaten wie Nummer, Baujahr und Material und seiner Lagegeometrie beschrieben, sondern auch durch seine grafische Darstellung - z.B. ein nicht ausgefülltes Quadrat mit der Kantenlänge von einem Meter, der Strichstärke 0.3 Millimeter, der Farbe Blau, etc. Diese grafische Darstellung kann in INTERLIS Version 2.0 ebenfalls modelliert werden.

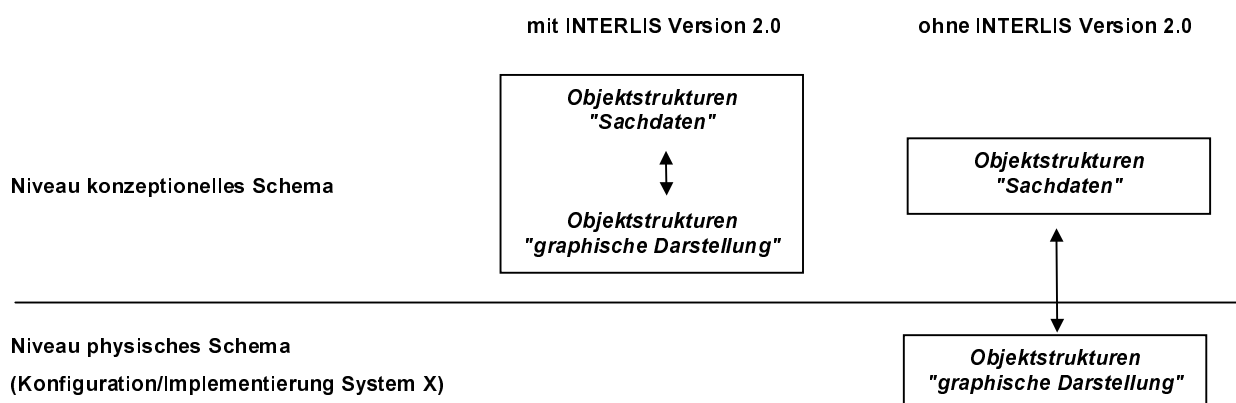


Abbildung 1: Vergleich Niveau grafische Darstellung mit und ohne INTERLIS Version 2.0

Die Möglichkeit der Beschreibung der grafischen Darstellung auf dem Niveau des konzeptionellen Schemas in INTERLIS Version 2.0 trägt entscheidend zu deren Transparenz bei. Die grafische Darstellung wird auf diese Weise detailliert dokumentiert, kann nachvollzogen werden und ist in maschinenlesbarer Form vorhanden. Ohne INTERLIS Version 2.0 muss die grafische Darstellung aus zum Teil systemspezifischen Konfigurationen und falls vorhanden, aus nicht formellen Dokumentationen eruiert werden.

## 2 Überblick Modelle, Daten, Grafik, Symbologie

INTERLIS Version 2.0 kennt diverse Arten von Modellen. Für die Beschreibung der grafischen Darstellung von Objekte sind die Modell-Arten **DATA/VIEW**, **GRAPHIC** und **SYMBOLGY** von Bedeutung.

In der Modell-Art **DATA MODEL** werden die Objektstrukturen der Daten definiert. Im Wesentlichen welche Themen und Objekte gibt es, welche Attribute und Beziehungen weisen die Objekte auf. Diese Modell-Art beinhaltet keine Definitionen zur grafischen Darstellung. Die Modell-Art **VIEW MODEL** ist ein Modell einer Sicht auf andere Modelle. Diese Modell-Art kann also auch ein Sicht auf eine Modell-Art **DATA MODEL** sein.

In der Modell-Art **SYMBOLGY MODEL** werden Signaturen für die grafische Darstellung definiert. Dass heisst, in dieser Modell-Art werden Symbole, Linienarten, Farben, Strichstärken etc. definiert. Die Definitionen innerhalb dieser Modell-Art weisen keine Referenzen auf die Objekte in der Modell-Art **DATA MODEL** respektive **VIEW MODEL** auf.

In der Modell-Art **GRAPHIC MODEL** werden im Wesentlichen die Referenzen zwischen den Objekten in der Modell-Art **DATA MODEL** respektive **VIEW MODEL** zu den Signaturen in der Modell-Art **SYMBOLGY MODEL** definiert. In dieser Modell-Art erfolgt die eigentliche Definition, mit welcher Signatur ein Objekt dargestellt werden muss.

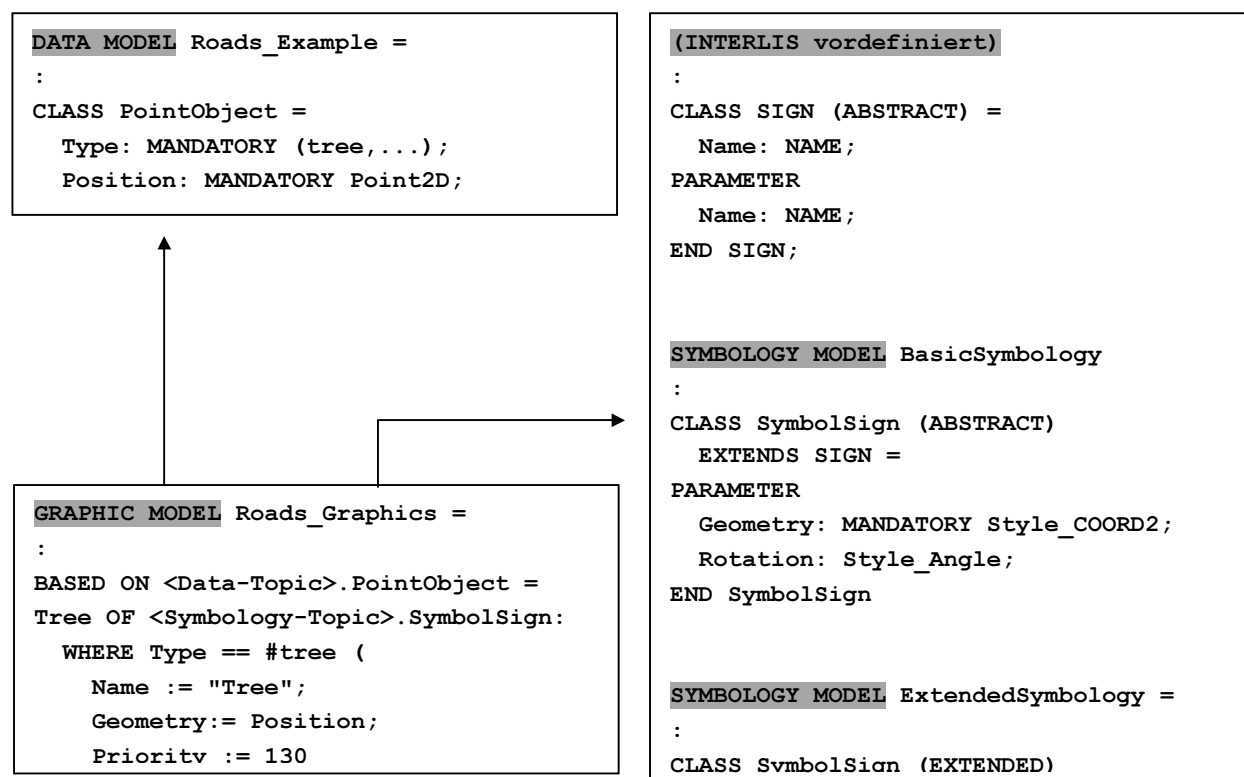


Abbildung 2: Übersicht Abhängigkeit DATA/VIEW MODEL, SYMBOLGY MODEL und GRAPHIC MODEL

Das Beispiel in der Abbildung 2 beschreibt im Datenmodell ein Punktobjekt in der Klasse `PointObject` mit dem Aufzählungsattribut `Type` mit einem möglichen Wert `#tree`.

Das Symbologiemodell beschreibt eine Symbolsignatur in der Klasse `SymbolSign` bestehend aus den Attributen `Name`, `Symbol`, `Scale`, `Color`, `Rotation`, `ClipSymbol` und den Parametern `Name`, `Priority` und `Rotation`.

Das Attribut `Name` und die Parameter `Geometry` und `Rotation` sind aus den Klassen `BasicSymbology.SymbolSign` und `SIGN` vererbt.

Im Grafikmodell erfolgt nun die Beschreibung der Darstellung des Punktobjektes vom `Type:=#tree` durch eine Referenz auf eine Symbolsignatur. Die Referenz auf die Symbolsignatur erfolgt über das Attribut `Name` der Klasse `SymbolSign`.

Das Punktobjekt vom `Type:=#tree` soll mit der Symbolsignatur `Name:="Tree"` und der `Priority:=130` dargestellt werden. Der Symbolsignatur wird über den Parameter `Geometry` die Lage des Punktobjektes aus dem Attribut `Position` übergeben.

Die Darstellung des Symbols ist in den Signatur-Attributen `Symbol`, `Scale`, `Color`, `Rotation` und `ClipSymbol` der Klasse `SymbolSign` definiert.

Das Attribut `Symbol` der Signatur definiert über die Referenz auf ein Symbol der Klasse `FontSymbol` die Geometrie des Symbols. Das Attribut `Scale` der Signatur definiert eine Skalierung des Symbols. Das Attribut `Color` der Signatur definiert eine Farbe in der Klasse `Color`. Das Attribut `Rotation` der Signatur definiert eine Rotation des Symbols. Das Attribut `ClipSymbol` der Signatur definiert einen Clip-Bereich für das Symbol mittels einem separaten Symbol in der Klasse `FontSymbol`. Der Parameter `Priority` der Signatur definiert über den Wert `130` die Priorität als numerischen Wert.

## 3 Die Symbologiemodelle in der Übersicht

### 3.1 Überblick

Die Symbologiemodelle, die im Rahmen der Möglichkeiten von INTERLIS 2 (Schlüsselwort `SYMBOLGY MODEL`) definiert wurden, beinhalten Signaturdefinitionen für Grafikobjekte wie Punkte, Linien, Flächen und Texte. Die Signaturdefinitionen für die Grafikobjekte beinhalten Signaturattribute wie Symbol, Linienart, Farbe, Strichdicke, etc. für die grafische Darstellung.

Dieses Dokument beschreibt die Definitionen von Signaturen basierend auf INTERLIS mit der hierarchischen Struktur der vordefinierten Klasse `SIGN` und den zwei Symbologiemodellen `BasicSymbology` und `ExtendedSymbology`.

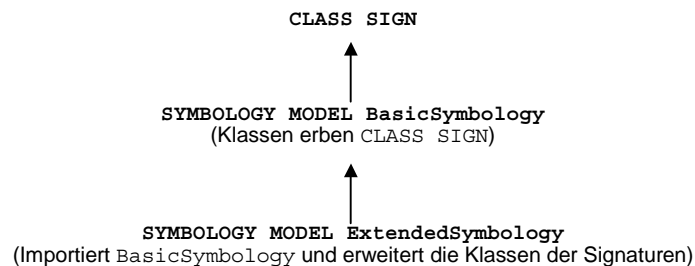


Abbildung 3: `CLASS SIGN`, `SYMBOLGY MODEL BasicSymbology` und `SYMBOLGY MODEL ExtendedSymbology`

### 3.2 Die vordefinierte `CLASS SIGN`

Für die Definition von Signaturen ist die Klasse `SIGN` durch INTERLIS vordefiniert. In Signatur-Klassen ist diese Basis-Klasse zu erben. Dadurch ist eine Signatur im Minimum durch einen Namen identifizierbar.

### 3.3 Das Symbologiemodell `BasicSymbology`

Das Symbologiemodell `BasicSymbology` enthält Basisdefinitionen zu Symbologien. Die Klassen in `BasicSymbology` erben die Klasse `SIGN`. Die Klassen in `BasicSymbology` definieren praktisch nur die Parameter aus den Daten, die der Signatur übergeben werden müssen, damit die Objekte an einer bestimmten Stelle erzeugt werden können. Die Klassen beinhalten noch keine konkreten Definitionen der Signatur wie Symbole, Farben, etc., die die Darstellung des Objektes beschreiben.

Das Symbologiemodell `BasicSymbology` ist das absolute Minimum, um Signaturen wenigstens benennen und die Parameter für die Erzeugung definieren zu können. Mit dem Symbologiemodell `BasicSymbology` alleine wäre die Darstellung der Objekte vollständig den Systemen überlassen. Mit dem Symbologiemodell `BasicSymbology` alleine können Systeme im Minimum anhand des Attributes Namen Signaturobjekte erkennen und selbständig umsetzen.

### 3.4 Das Symbologiemodell `ExtendedSymbology`

Das Symbologiemodell `ExtendedSymbology` erweitert das Symbologiemodell `BasicSymbology`. Die Klassen in `ExtendedSymbology` erweitern die entsprechenden Klassen in `BasicSymbology` um konkrete Definitionen der Signatur wie Symbole, Farben etc. Für die Definitionen der Signaturen enthält `ExtendedSymbology` zudem weitere Strukturen und Klassen, um Werte von Signaturen wie Farben, Symbole, etc. in der Form von Bibliotheken beschreiben zu können.

## 4 Die Symbologiemodelle im Detail

### 4.1 Einleitung

In den nachfolgenden Kapiteln wird das diesem Dokument zugrunde liegende Symbologiemodell detailliert beschrieben. Die Anwendung dieses Symbologiemodelles im Zusammenhang mit dem Datenmodell und dem Grafikmodell kann im Kapitel "Ein konkretes Beispiel" betrachtet werden.

### 4.2 Die vordefinierte CLASS SIGN

Für die Signaturen ist in INTERLIS die Klasse `SIGN` vordefiniert. Alle Signatur-Klassen müssen diese Klasse erben. Damit ist gewährleistet, dass eine Signatur im Minimum über einen Namen identifiziert werden kann.

<b>Klasse</b>	<pre> CLASS SIGN (ABSTRACT) =   Name: NAME; PARAMETER   Name: NAME; END SIGN; </pre>
<b>Beschreibung</b>	Durch die Vererbung dieser Klasse in den Klassen für Signaturen, enthält eine Signatur im Minimum einen Namen.
<b>Beispiel</b>	Die Signatur <code>Tree</code> wird definiert. <pre> Name := "Tree"; </pre>

### 4.3 Das Symbologiemodell `BasicSymbology`

Das Symbologiemodell `BasicSymbology` enthält Basis-Klassen für die Signaturen. Für jedes gebräuchliche Darstellungsobjekt wie Text, Symbol, Polylinie und Fläche ist eine Klasse definiert.

<b>Klassen</b>	<pre> CLASS TextSign (ABSTRACT) EXTENDS SIGN =   Name      : NAME; PARAMETER   Name      : NAME;   Txt       : MANDATORY TEXT;   Geometry  : MANDATORY Style COORD2;   Rotation  : Style ANGLE;   HAli     : HALIGNMENT;   VAlI     : VALIGNMENT; END TextSign;  CLASS SymbolSign (ABSTRACT) EXTENDS SIGN =   Name      : NAME; PARAMETER   Name      : NAME;   Geometry  : MANDATORY Style_COORD2;   Rotation  : Style_ANGLE; END SymbolSign;  CLASS PolylineSign (ABSTRACT) EXTENDS SIGN =   Name      : NAME; PARAMETER   Name      : NAME;   Geometry  : MANDATORY Style_POLYLINE; END PolylineSign;  CLASS SurfaceSign (ABSTRACT) EXTENDS SIGN =   Name      : NAME; PARAMETER   Name      : NAME;   Geometry  : MANDATORY Style_SURFACE; END SurfaceSign;  Kursive Attribute oder Parameter stammen aus der in INTERLIS vordefinierten Klasse SIGN.</pre>
<b>Beschreibung</b>	<p>Die Klassen erweitern die Klasse <code>SIGN</code>, die nur das Attribut <code>Name</code> enthält. Die Klassen definieren lediglich die minimalen Parameter für das Grafikmodell, welches die Schnittstelle zwischen Daten im Datenmodell und den Signaturen im Symbologiemodell definiert. Durch die Parameter kann im Grafikmodell zum Beispiel für ein Punktobjekt der Namen, die Position und die Ausrichtung des Datenobjektes an das Signaturobjekt übergeben werden.</p>
<b>Beispiel</b>	<p>Die Symbolsignatur <code>Tree</code> wird definiert.</p> <pre> Name      := "Tree"; Geometry  := 600000.0/200000.0; Rotation  := 30.0;</pre>

### 4.4 Das Symbologiemodell `ExtendedSymbology`

#### 4.4.1 Einleitung

Im Symbologiemodell `BasicSymbology` wurden keine Definitionen zur grafischen Darstellung der Signaturen definiert. Diese Definitionen erfolgen nun im Symbologiemodell `ExtendedSymbology`.

Die Klassen im Symbologiemodell `ExtendedSymbology` können wie folgt unterteilt werden:

##### Klassen für Bibliotheken

Beispiel: `CLASS Color`

Diese Klassen beinhalten Definitionen, auf die mittels Beziehungsattributen in den Klassen für die Signaturen referenziert wird.

Zum Beispiel wird in der Signaturklasse `SymbolSign` auf die die Klasse `Color` verwiesen.

Diese Klassen definieren Wertebereiche für die Attribute in den Klassen für die Signaturen.

##### Klassen für Signaturen

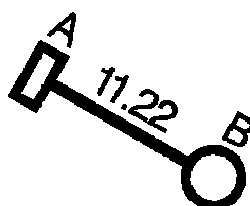
Beispiel: `CLASS SymbolSign`

Diese Klassen definieren für das Grafikmodell die Signaturen. Im Grafikmodell kann für Datenobjekte eine Referenz auf diese Signaturobjekte für die grafische Darstellung definiert werden. Die Signatur definiert, mit welcher Geometrie, Farbe, etc. das Datenobjekt dargestellt werden soll.

In den nachfolgenden Kapiteln werden die einzelnen Strukturen und Klassen des Symbologiemodelles `ExtendedSymbology` näher beschrieben.

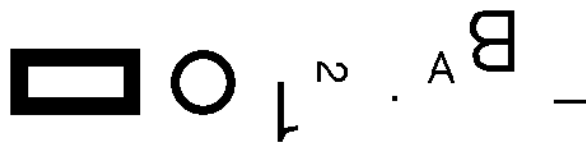
#### 4.4.2 Geometrische Ausdehnungen/Ausrichtungen

Die Definitionen im Symbologiemodell `ExtendedSymbology` beinhalten eine Reihe von Attributen, deren Werte die geometrische Ausdehnungen/Ausrichtungen - Höhe, Breite, Rotation, etc. - der Signaturen beschreiben. Um die Werte dieser Attribute in den verschiedenen Signaturen in einer vernünftigen geometrischen Relation zueinander definieren zu können, werden geometrische Grundsätze vorausgesetzt.



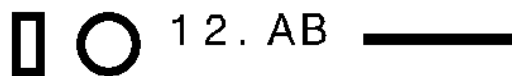
*Abbildung 3: Beispiel Darstellung. Zwei Symbole mit Beschriftungen sind durch eine Polyline mit einer Beschriftung verbunden. Die geometrische Ausdehnung und Ausrichtung der einzelnen Signaturen der Symbole, Texte und Linie müssen in der geforderten geometrischen Relation zueinander dargestellt werden.*

Grundsätzlich können die Signaturen für die in Abbildung 3 vorkommenden Signaturen ohne gegenseitige geometrische Relation definiert werden. Dies führt aber dazu, dass die einzelnen Signaturen unterschiedlich in der Skalierung und Rotation für die Darstellung abgehandelt werden müssen.



*Abbildung 4: Die Signaturen für das Beispiel in Abbildung 3 sind nicht in gegenseitiger geometrischer Relation definiert. Um mit diesen Signaturen die Darstellung in Abbildung 3 zu erreichen, ist jede einzelne Signatur separat zu skalieren und zu rotieren.*

Besser ist es, die Signaturen in gegenseitiger geometrischen Relation zu definieren. Dadurch können die Signaturen mit einer einheitlichen Skalierung und Rotation für die Darstellung behandelt werden. In Kommentaren zu den Definitionen der Signaturen können weitere informelle Beschreibungen mitgegeben werden. Zum Beispiel für welche Planmassstäbe und für welche Thematik die Signaturen definiert sind - z.B. 1:500/1:1000, Hydrantenplan.



*Abbildung 5: Die Signaturen für das Beispiel in Abbildung 3 sind in gegenseitiger geometrischer Relation definiert. Um mit diesen Signaturen die Darstellung in Abbildung 3 zu erreichen, können alle Signaturen mit der identischen Skalierung und Rotation plaziert werden.*

Um die Signaturen in gegenseitiger geometrischer Relation definieren zu können, sind folgende geometrischen Grundsätze einzuhalten.

### **Globale Ausrichtung**

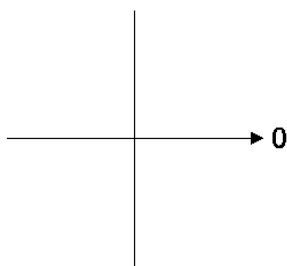


Abbildung 5: Globale Ausrichtung.

Die globale Ausrichtung für die Signaturen ist definiert durch:

- Die Ostrichtung ist die Nullrichtung (Winkel = 0)
- Die Winkeleinheit ist Altgrad

Für die Signaturen ist die Ostrichtung als Nullrichtung gewählt, um die Definition der Geometrien der Signaturen zu erleichtern. So können zum Beispiel Symbole von Symbol- und Textfonts in ihrer gebräuchlichen, horizontalen Ausrichtung definiert werden.

Ist im Datenmodell für Winkelangaben eine andere Nullrichtung definiert, zum Beispiel die Nordrichtung, so sind die Signaturen entsprechend zu transformieren.

### Globales Koordinatensystem/Einheit

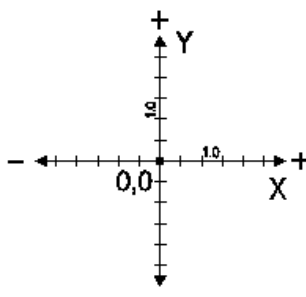


Abbildung 6: Globales Koordinatensystem/Einheit.

Das globale Koordinatensystem und die globale Einheit für die Signaturen ist definiert durch:

- Die X-Achse ist horizontal und von Westen nach Osten aufsteigend.
- Die Y-Achse ist vertikal und von Süden nach Norden aufsteigend.
- Die X- und Y-Achse stehen rechtwinklig zueinander.
- Der Ursprung des Koordinatensystemes beträgt  $X=0.0$  und  $Y=0.0$  beim Schnittpunkt der Achsen.
- Die Einheit beträgt 1.0  
Die Einheit kann in Meter als Benutzereinheit betrachtet werden.

### Ausdehnung/Ausrichtung für Symbole in Symbolfonts und Textfonts

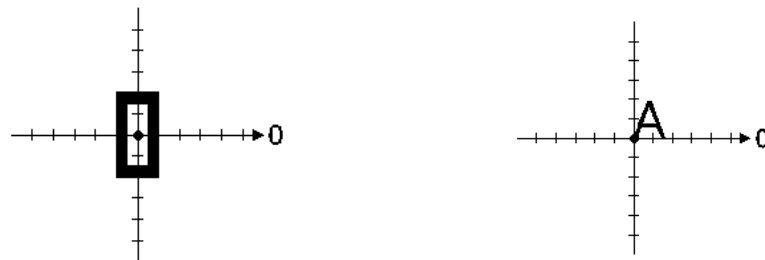


Abbildung 7: Ausdehnung/Ausrichtung für Symbole in Symbolfonts und Textfonts.

Links: Beispiel für ein Symbol in einem Symbolfont. Rechts: Beispiel für ein Symbol in einem Textfont.

Die Ausdehnung/Ausrichtung von Signaturen für Symbole in Symbolfonts ist definiert durch:

- Die Ausrichtung ist gegen Osten, respektive 0 Altgrad.
- Der Einfügepunkt für das Symbol ist die Koordinate  $X/Y=0.0/0.0$
- Die Geometrien des Symbols sind relativ zu der globalen Einheit=1.0 zu definieren.
- Skalierungen der Geometrien des Symbols erfolgen um den Einfügepunkt
- Rotationen der Geometrien des Symbols erfolgen um den Einfügepunkt
- Der Einfügepunkt von Symbolen für Textfonts ist die untere, linke Ecke der geometrischen Ausdehnung des Symbols.

Wird zum Beispiel ein Symbol für ein Objekt mit der Lage  $X/Y=600000.000/200000.00$ , einer Skalierung von 2.0 und einem Winkel von 120.0 Grad (in DEGREES, Nullrichtung ist die Nordrichtung) platziert, so muss das Symbol mit dem Einfügepunkt an der Lage  $X/Y=600000.000/200000.00$  platziert werden, die Geometrien des Symbols um den Einfügepunkt mit der Skalierung=2.0 skaliert werden und die Geometrien des Symbols um den Einfügepunkt mit dem Winkel = 30.0 Grad (-90.0 Grad zur Nullrichtung im Norden, +120.0 Grad Rotation) rotiert werden.

### Ausdehnung/Ausrichtung für Linienarten



Abbildung 8: Ausdehnung/Ausrichtung für Linienarten.

Links: Beispiel für eine strichlierte (dashed) Linienart. Rechts: Beispiel für eine bemusterte (pattern) Linienart.







Die Ausdehnung/Ausrichtung von Signaturen für Linienarten ist definiert durch:

- Die Ausrichtung ist gegen Osten, respektive 0 Altgrad.
- Der Startpunkt für Distanzen ist die Koordinate  $X/Y=0.0/0.0$ .
- Distanzangaben erfolgen mit positiven Werten in der X-Richtung.
- Offsetangaben erfolgen rechtwinklig zur X-Richtung.
- Distanzen und Offsets für Linienarten sind relativ zu der globalen Einheit=1.0 zu definieren.
- Skalierungen erfolgen um den Startpunkt.

### 4.4.3 Farb-Bibliothek

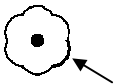

<b>Klasse</b>	<pre> CLASS Color =   Name : TEXT*40;   L    : MANDATORY 0.0 .. 100.0;   C    : MANDATORY 0.0 .. 181.1;   H    : MANDATORY 0.0 .. 359.9;   T    : MANDATORY 0.000 .. 1.000; END Color; </pre>
<b>Beschreibung</b>	Die Klasse <code>Color</code> definiert Farben, auf die in den Farbattributen in den Signaturklassen verwiesen wird.
<b>Attribut <code>Name</code></b>	Jeder Farbe kann über das Attribut <code>Name</code> ein Namen vergeben werden.
<b>Attribut <code>L, C, H</code></b>	Definiert die Farbe wird im Farbraum $L^*C^*_{ab} h^*_{ab}$ über die Attribute <code>L</code> : Luminance, <code>C</code> : Chroma, <code>H</code> : Hue. Detaillierte Angaben über den Farbraum $L^*C^*_{ab} h^*_{ab}$ siehe in [2].
<b>Attribut <code>T</code></b>	Definiert die Transparenz der Farbe von <code>0.000</code> = durchsichtig bis <code>1.000</code> = deckend definiert.
<b>Beispiel</b>	<p>Die Farbe Braun ohne Transparenz wird über folgende Werte definiert:</p> <pre> Name := "Braun"; L    := 50.0; C    := 58.3; H    := 59.0; T    := 1.000; </pre>

### 4.4.4 Polylinienattribut-Bibliothek

<b>Klasse</b>	<pre> CLASS PolylineAttrs =   Width      : ES Float;   Join       : (bevel,round,miter);   MiterLimit : 1.0 .. 1000.0;   Caps       : (round,butt); END PolylineAttrs;         </pre>
<b>Beschreibung</b>	<p>Die Klasse <code>PolylineAttrs</code> definierte Basis-Darstellungsattribute für eine einfache durchgezogene Linie.</p>
<b>Attribut <code>width</code></b>	<p>Strichbreite der Linie.</p>
<b>Attribut <code>Join</code></b>	<p>Verbindungsart von Zwischenpunkten der Linie.</p> <div style="display: flex; justify-content: space-around; text-align: center;"> <div data-bbox="614 792 743 1043"> <p>bevel</p>  </div> <div data-bbox="847 792 976 1043"> <p>round</p>  </div> <div data-bbox="1074 792 1203 1043"> <p>miter</p>  </div> </div>
<b>Attribut <code>MiterLimit</code></b>	<p>Limite für die Verbindungsart <code>miter</code> der Zwischenpunkte. In der Verbindungsart <code>miter</code> werden die äusseren Ränder der beiden Linien verlängert, bis sie sich kreuzen. Mit dem Attribut <code>MiterLimit</code> wird diese Verlängerung limitiert, die sonst bei fast parallelen Linien ins quasi Unendliche führen kann. Wird bei der Verlängerung die <code>MiterLimit</code> erreicht, werden die beiden Linien in der Linienart <code>bevel</code> verbunden.</p>
<b>Attribut <code>Caps</code></b>	<p>Abschluss am Ende der Linie.</p> <div style="display: flex; justify-content: space-around; text-align: center;"> <div data-bbox="703 1541 810 1637"> <p>round</p>  </div> <div data-bbox="1010 1541 1090 1637"> <p>butt</p>  </div> </div>
<b>Beispiel</b>	<p>Folgende Linie wird definiert.</p> <pre> Width      := 2.500; Join       := miter; MiterLimit := 2.500; Caps       := round;         </pre> <div style="text-align: center; margin-top: 20px;">  </div>

### 4.4.5 Font-Bibliothek

Mit den Strukturen und Klassen der Fontbibliothek werden Symbolbibliotheken für die Signaturen definiert. Eine Font besteht aus Symbolen, welche wiederum aus Geometrien bestehen. Ein Font kann ein Textfont oder ein Symbolfont sein.

<b>Strukturen</b>	<pre> STRUCTURE FontSymbol Geometry (ABSTRACT) = END FontSymbol_Geometry;  STRUCTURE FontSymbol Polyline EXTENDS FontSymbol_Geometry =   Color      : -&gt; Color;   LineAttrs  : -&gt; PolylineAttrs;   Geometry   : MANDATORY ES_Polyline; END FontSymbol_Polyline;  STRUCTURE FontSymbol_Surface EXTENDS FontSymbol_Geometry =   FillColor  : -&gt; Color; !! nur fuer Symbole   Geometry   : MANDATORY ES_Surface; END FontSymbol_Surface; </pre>
<b>Beschreibung</b>	<p>Die aufgeführten Strukturen beschreiben ein atomares geometrisches Element eines einzelnen Symbols. Diese Strukturen werden bei der Klasse für die Symbole benötigt, um ein Symbol geometrisch beschreiben zu können. Die aufgeführten Geometrien können nicht selbständig auftreten, sondern sind in einer Liste eines Symbols enthalten.</p> <p>Die Struktur <code>FontSymbol_Geometry</code> ist lediglich eine Basisstruktur zur einheitlichen Behandlung von Symbolgeometrien. Sie wird von den nachfolgenden Strukturen geerbt.</p> <p>Die Struktur <code>FontSymbol_Polyline</code> definiert eine atomare Polyline eines Symbols.</p> <p>Die Struktur <code>FontSymbol_Surface</code> definiert eine atomare Fläche eines Symbols.</p>
<b>Attribut <code>Color</code></b>	Definiert die Farbe einer Polyline.
<b>Attribut <code>FillColor</code></b>	Definiert die Füllfarbe einer Fläche.
<b>Attribut <code>Geometry</code></b>	Definiert die Geometrie der Polyline, respektive der Fläche.
<b>Beispiel</b>	<div style="text-align: center;">  </div> <p>Entspricht einer <code>FontSymbol_Polyline</code> des Symbols.</p> <div style="text-align: center;">  </div> <p>Entspricht einer <code>FontSymbol_Surface</code> des Symbols.</p>

Das Beispiel stellt ein Symbol dar. Dieselben Aussagen gelten ebenfalls für Symbole eines Textfonts, wobei ein Symbol in einem Textfont einem Textzeichen entspricht.

<b>Klasse</b>	<pre> CLASS FontSymbol =   Name      : TEXT*40;   UCS4      : 0 .. 4000000000;   Spacing   : ES Float;   Geometry  : LIST OF FontSymbol Geometry; END FontSymbol;         </pre>
<b>Beschreibung</b>	Die Klasse <code>FontSymbol</code> definiert ein Symbol einer Fontbibliothek.
<b>Attribut <code>Name</code></b>	Definiert den Namen des Symbols.
<b>Attribut <code>ucs4</code></b>	Anwendung nur bei Symbolen eines Textfonts. Definiert der dem Textzeichen entsprechende UCS4-Wert (Unicode).
<b>Attribut <code>spacing</code></b>	Anwendung nur bei Symbolen eines Textfonts. Definiert den Abstand des Textzeichens zu einem nachfolgenden Zeichen.
<b>Attribut <code>Geometrie</code></b>	<p>Definiert über eine geordneten Liste von atomaren Geometrien - Polyline und/oder Flächen - die gesamte Geometrie des Symbols.</p> <p>Die Reihenfolge in der Liste bestimmt die Hierarchie der einzelnen Geometrien. Die erste Geometrie der Liste wird als erste dargestellt, die letzte Geometrie der Liste wird als letzte dargestellt. Jeder nachfolgende Geometrie liegt über der vorhergehenden Geometrie.</p> <p>Der Einfügepunkt des Symbols entspricht der Koordinate 0.0/0.0. Alle Geometrien des Symbols sind relativ zu diesem Einfügepunkt zu definieren.</p> <p>Alle Symbole eines Fonts sollten in der Skalierung über ihre geometrischen Ausdehnung aufeinander abgestimmt sein. Dies betrifft ebenfalls den Wert des Attributes <code>Spacing</code>.</p>
<b>Beispiel</b>	<p>Es wird das Textzeichen "A" als Symbol für einen Textfont definiert.</p> <div style="text-align: center;"> </div> <pre> Name      := "A"; UCS4      := 65; Spacing   := 1.200; Geometry  := Liste von Referenzen auf Instanzen             der Klasse FontSymbol_Geometry;         </pre>

<b>Klasse</b>	<pre> CLASS Font =   Name      : MANDATORY TEXT*40;   Internal  : MANDATORY BOOLEAN;   Type      : MANDATORY (symbol, text);   BottomBase : ES Float;   Symbols   : LIST OF FontSymbol; END Font; </pre>
<b>Beschreibung</b>	Die Klasse <code>Font</code> gruppiert eine logische zusammengehörende Menge von Symbolen als Liste zu einem Font.
<b>Attribut <code>Name</code></b>	Definiert den Namen des Fonts.
<b>Attribut <code>Internal</code></b>	<p>Falls die Symbole des Fonts in der Liste des Attributes <code>Symbols</code> definiert sind, handelt es sich um einen internen Font - <code>Internal := True</code>.</p> <p>Falls die Symbole des Fonts nicht in der Liste des Attributes <code>Symbols</code> definiert sind, handelt es sich um einen externen Font - <code>Internal := False</code>.</p> <p>In diesem Falle enthält der Font keine konkreten Symbole. Über das Attribut <code>Name</code> des Fonts wird lediglich ein Font benannt.</p>
<b>Attribut <code>Type</code></b>	Definiert den Type des Fonts - <code>symbol</code> oder <code>text</code> .
<b>Attribut <code>BottomBase</code></b>	<p>Definiert einen globalen unteren Abstand (Höher-/Tieferstellen) aller Symbole des Fonts zum jeweiligen Einfügepunkt des einzelnen Symbols.</p> <div style="text-align: center;"> </div>
<b>Attribut <code>Symbols</code></b>	Enthält die Symbole des Fonts in der Form einer Liste mit Objekten der Klasse <code>FontSymbol</code> .
<b>Beispiel</b>	<p>Definiert wird der Font <code>Arial</code>.</p> <pre> Name      := "Arial"; Internal  := True; Type      := text; BottomBase := 0.0; Symbols   := Liste von Referenzen auf Instanzen             der Klasse FontSymbol; </pre>

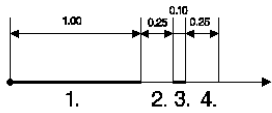
#### 4.4.6 Linienarten-Bibliothek

Mit den Strukturen und Klassen der Linienarten-Bibliothek werden Linienarten definiert. Die Linienarten können ausgezogene, strichlierte und durch Symbole bemusterte Linien sein. Eine Linienart kann auch Bestandteil einer aus mehreren Linienarten bestehenden Liniensignatur (Multiline) sein.

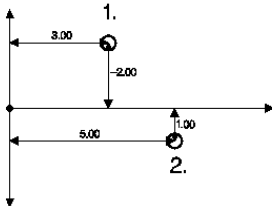
<b>Klasse</b>	<pre>CLASS LineStyle (ABSTRACT) =   Name : MANDATORY TEXT*40; END LineStyle;</pre>
<b>Beschreibung</b>	Die Klasse <code>LineStyle</code> ist die Basisklasse zur einheitlichen Behandlung von Linienarten. Sie wird von den nachfolgenden Klassen <code>LineStyle_Solid</code> , <code>LineStyle_Dashed</code> und <code>LineStyle_Pattern</code> geerbt.
<b>Attribut Name</b>	Definiert den Namen der Linienart.
<b>Beispiel</b>	Definiert wird die Linienart <code>solid</code> .  Name := "Solid";

<b>Klasse</b>	<pre>CLASS LineStyle_Solid EXTENDS LineStyle =   Offset      : ES_Float;   Color       : -&gt; Color;   LineAttrs   : -&gt; PolylineAttrs; END LineStyle_Solid;</pre>
<b>Beschreibung</b>	Mit der Klasse <code>LineStyle_Solid</code> werden ausgezogene Linienarten definiert.
<b>Attribut Offset</b>	Definiert den Offset der Linie zur Achse der Basislinie. Falls die Liniensignatur, der diese Linienart zugehört, aus einer einzigen Linie besteht, beträgt der Wert normalerweise <code>Offset := 0.0</code> und entspricht der Achse der Linie. Falls die Liniensignatur, der diese Linienart zugehört, aus mehreren Linien besteht, wird mit dem Wert <code>Offset</code> der Offset zur Achse definiert. Ein Minuswert entspricht einem linken Offset in Laufrichtung der Linie gesehen. Ein Pluswert entspricht einem rechten Offset in Laufrichtung der Linie gesehen.
<b>Attribut Color</b>	Definiert die Farbe der Linienart.
<b>Attribut LineAttrs</b>	Definiert die Linienattribute der Linienart.
<b>Beispiel</b>	Definiert wird die Linienart "linke Wand" einer Liniensignatur für eine Leitung, die aus drei Linien besteht - der Achse, der linken Wand und der rechten Wand.  Name := "Leitung linke Wand"; Offset := -1.0;

	<pre> Color          := Referenz auf eine Instanz                   der Klasse Color, z.B. "Blau";  LineAttrs      := Referenz auf eine Instanz                   der Klasse PolylineAttrs; </pre>
<b>Struktur/Klasse</b>	<pre> STRUCTURE DashRec =   DLength      : ES_Float; END DashRec;  CLASS LineStyle_Dashed EXTENDS LineStyle =   Offset       : ES_Float;   Color        : -&gt; Color;   LineAttrs    : -&gt; PolylineAttrs;   Dashes       : LIST OF DashRec; END LineStyle_Dashed; </pre>
<b>Beschreibung</b>	Mit der Klasse <code>LineStyle_Dashed</code> werden gestrichelte Linienarten definiert.
<b>Attribut <code>Offset</code></b>	<p>Definiert den Offset der Linie zur Achse der Basislinie.</p> <p>Falls die Liniensignatur, der diese Linienart zugehört, aus einer einzigen Linie besteht, beträgt der Wert normalerweise <code>Offset := 0.0</code> und entspricht der Achse der Linie. Falls die Liniensignatur, der diese Linienart zugehört, aus mehreren Linie besteht, wird mit dem Wert <code>Offset</code> der Offset zur Achse definiert.</p> <p>Ein Minuswert entspricht einem linken Offset in Laufrichtung der Linie gesehen. Ein Pluswert entspricht einem rechten Offset in Laufrichtung der Linie gesehen.</p>
<b>Attribut <code>Color</code></b>	Definiert die Farbe der Linienart.
<b>Attribut <code>LineAttrs</code></b>	Definiert die Linienattribute der Linienart.
<b>Attribut <code>Dashes</code></b>	<p>Definiert eine geordnete Liste von Teilstrichen als Linienart.</p> <p>Der 1. Teilstrich der Liste wird dargestellt.</p> <p>Der 2. Teilstrich der Liste wird nicht dargestellt, respektive entspricht dem Abstand zum nächsten dargestellten Teilstrich.</p> <p>Der 3. Teilstrich der Liste wird dargestellt.</p> <p>Der 4. Teilstrich der Liste wird nicht dargestellt, respektive entspricht dem Abstand zum nächsten dargestellten Teilstrich.</p> <p>Und so weiter.</p>
<b>Beispiel</b>	<p>Definiert wird die Linienart "Achse".</p> <pre> Name          := "Achse"; Offset        := 0.0; Color         := Referenz auf eine Instanz                   der Klasse Color, z.B. "Blau";  LineAttrs     := Referenz auf eine Instanz                   der Klasse PolylineAttrs;  Dashes        := Liste von geordneten Referenzen auf Instanzen </pre>

	<p>der Struktur DashRec;</p> <p>Die Liste der Instanzen in DashRec sieht wie folgt aus:</p>  <p>Der 1. Eintrag 1.00 in der Liste definiert die Länge des ersten dargestellten Teilstriches.</p> <p>Der 2. Eintrag 0.25 in der Liste definiert die nicht dargestellte Länge zum nächsten dargestellten Teilstrich.</p> <p>Der 3. Eintrag 0.10 in der Liste definiert die Länge des nächsten dargestellten Teilstriches.</p> <p>Der 4. Eintrag 0.25 in der Liste definiert die nicht dargestellte Länge zum nächsten dargestellten Teilstrich. Da die Liste hier zu Ende ist, wiederholt sich die Abfolge der Liste wieder.</p>
--	---

<b>Struktur/Klasse</b>	<pre>STRUCTURE Pattern Symbol =   Symbol      : MANDATORY -&gt; FontSymbol;   Weight      : ES Float;   Scale       : ES Float;   Color       : -&gt; Color;   Dist        : MANDATORY ES Float;   Offset      : MANDATORY ES Float; END Pattern Symbol;  CLASS LineStyle Pattern EXTENDS LineStyle =   Offset      : ES Float;   PLength     : MANDATORY ES Float;   Symbols     : LIST OF Pattern Symbol; END LineStyle Pattern;</pre>
<b>Beschreibung</b>	<p>Mit der Klasse <code>LineStyle_Pattern</code> werden bemusterte Linienarten definiert. Die bemusterte Linienart kann mehrere Bemusterungs-Symbole beinhalten.</p> <p>Mit der Struktur <code>Pattern_Symbol</code> werden die Symbole für die Bemusterung definiert.</p>
<b>Attribut</b> <code>Pattern_Symbol</code> <code>Symbol</code>	Definiert das Symbol für die Bemusterung.
<b>Attribut</b> <code>Pattern_Symbol</code> <code>Weight</code>	Definiert die Strichdicke für die Geometrien des Symbols.
<b>Attribut</b> <code>Pattern_Symbol</code>	Definiert die Skalierung des Symbols. Default = 1.0 .

<b>Scale</b>	
<b>Attribut</b> <b>Pattern_Symbol</b> <b>Color</b>	Definiert die Farbe des Symbols. Falls definiert, werden die Farben der einzelnen Geometrien des Symbols übersteuert.
<b>Attribut</b> <b>Pattern_Symbol</b> <b>Dist</b>	Definiert die Distanz vom Bemusterungsursprung der Linienart zum Ursprung des Symbols entlang der Linie.
<b>Attribut</b> <b>Pattern_Symbol</b> <b>Offset</b>	Definiert den Offset von der Bemusterungsachse der Linienart zum Ursprung des Symbols. Ein Minuswert entspricht einem linken Offset entlang der Linie gesehen. Ein Pluswert entspricht einem rechten Offset entlang der Linie gesehen.
<b>Attribut</b> <b>offset</b>	Definiert den Offset der Linie zur Achse der Basislinie. Falls die Liniensignatur, der diese Linienart zugehört, aus einer einzigen Linie besteht, beträgt der Wert normalerweise <code>offset := 0.0</code> und entspricht der Achse der Linie. Falls die Liniensignatur, der diese Linienart zugehört, aus mehreren Linie besteht, wird mit dem Wert <code>offset</code> der Offset zur Achse definiert. Ein Minuswert entspricht einem linken Offset in Laufrichtung der Linie gesehen. Ein Pluswert entspricht einem rechten Offset in Laufrichtung der Linie gesehen.
<b>Attribut</b> <b>Plength</b>	Definiert die Länge, nach der die Bemusterung der Linienart wiederholt wird.
<b>Attribut</b> <b>symbols</b>	Definiert eine geordnete Liste von Symbolen für die Bemusterung der Linienart.
<b>Beispiel</b>	<p>Definiert wird die Linienart "Pattern".</p> <pre>Name           := "Pattern"; Plength := 5.0; Symbols        := Liste von geordneten Referenzen                 auf Instanzen der Struktur                 Pattern_Symbol;</pre> <p>Die Liste der Instanzen in <code>Pattern_Symbol</code> sieht wie folgt aus:</p>  <p>Der 1. Eintrag in der Liste definiert ein Symbol mit der Distanz 3.0 und dem Offset -2.0 vom Startpunkt der Linie. Der 2. Eintrag in der Liste definiert ein Symbol mit der Distanz 5.0 und dem Offset 1.0 vom Startpunkt der Linie.</p>

--	--

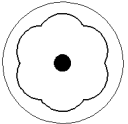
#### 4.4.7 Textsignatur

<b>Klasse</b>	<pre> CLASS TextSign (EXTENDED) =   Name:      NAME;   Font       : MANDATORY -&gt; Font;   Height    : MANDATORY ES Float;   Weight    : ES Float;   Slanted   : BOOLEAN;   Underlined : BOOLEAN;   Striked   : BOOLEAN;   Color     : -&gt; Color;   ClipBox   : ES Float;   ClipFont  : -&gt; Font; MANDATORY CONSTRAINT   Font.Type == #text; PARAMETER   Name      : NAME;   Txt       : MANDATORY TEXT;   Geometry  : MANDATORY Style COORD2;   Rotation  : Style ANGLE;   HAli     : HALIGNMENT;   VAlI     : VALIGNMENT;   Priority  : MANDATORY ES Priority; END TextSign; </pre> <p>Kursive Attribute oder Parameter stammen aus gleichnamigen Klasse im Symbologiemodell BasicSymbology.</p>
<b>Beschreibung</b>	<p>Mit der Klasse <code>TextSign</code> werden Textsignaturen, respektive Characters für Fonts definiert.</p> <p>Die Klasse <code>TextSign</code> erweitert die gleichnamige Klasse aus dem SYMBOLOGY MODEL <code>BasicSymbology</code> um effektive Darstellungsattribute, unter anderen auch um Attribute mit Werten aus den vorgängig definierten Signaturbibliotheken wie <code>Color</code>, etc.</p>
<b>Attribut <code>Font</code></b>	<p>Definiert den Font aus der Fontbibliothek für die Signatur.</p>
<b>Attribut <code>Height</code></b>	<p>Definiert die Höhe, respektive die Skalierung für die Symbole des Fonts. Die Geometrien der einzelnen Symbole des Fonts werden um diesen Wert skaliert.</p>
<b>Attribut <code>Weight</code></b>	<p>Definiert die Strichdicke der Geometrien für die Symbole des Fonts.</p>
<b>Attribut <code>slanted</code></b>	<p>Sollen die Symbole des Fonts schräggestellt sein: <i>ABC</i>.</p> <p>Die Darstellung der Schrägstellung wird dem System freigelassen.</p>

<b>Attribut Underlined</b>	<p>Sollen die Symbole des Fonts unterstrichen werden: <u>ABC</u> . Die Darstellung der Unterstreichung wird dem System freigelassen.</p>
<b>Attribut Striked</b>	<p>Sollen die Symbole des Fonts durchstrichen werden: <del>ABC</del> . Die Darstellung der Durchstreichung wird dem System freigelassen.</p>
<b>Attribut Color</b>	<p>Definiert die Farbe des Symbole des Fonts. Falls definiert, übersteuert der Wert die Farbe bei den einzelnen Symbolen.</p>
<b>Attribut ClipBox</b>	<p>Definiert eine rechteckigen Box um einen Text des Fonts, die der maximalen Ausdehnung des Texts plus dem Wert von ClipBox entspricht.</p> <div data-bbox="837 801 1098 922" data-label="Image"> </div> <p>Ein Clipping hat den Hintergrund um diese Box auszuschneiden.</p>
<b>Attribut ClipFont</b>	<p>Definiert den Clippfont aus der Fontbibliothek für die Signatur. Jedes Symbol des Fonts definiert mit dem Attribut Font hat eine Entsprechung im Font definiert mit dem Attribut ClipFont . Die Entsprechung erfolgt über den identischen UCS4-Code des Symbols. Das Symbol im Clippfont definiert eine Fläche für das Symbol des Font.</p> <div data-bbox="887 1339 1042 1415" data-label="Image"> </div> <p>Ein Clipping hat den Hintergrund um dieses Clippsymbol pro Symbol auszuschneiden.</p> <p>Falls das Attribut ClipFont auf einen Font verweist, wird das Attribut ClipBox ignoriert.</p>
<b>Constraint</b> Font.Type=#text	<p>Deklariert den Font als einen Textfont.</p>
<b>Parameter Priority</b>	<p>Definiert die Priorität der Signatur. Eine Priorität mit einem höheren Wert - z.B. 5 - "liegt" über einer Priorität mit einem tieferen Wert - z.B. 4 - .</p>
<b>Beispiel</b>	<p>Definiert wird die Textsignatur "Parzellennummer".</p>

	<p><b>Attribute:</b></p> <p>Name := "Parzellennummer";</p> <p>Font := Referenz auf eine Instanz der Klasse Font, z.B. "Arial";</p> <p>Height := 1.0;</p> <p>Weight := 0.01;</p> <p>Slanted := False;</p> <p>Underlined := False;</p> <p>Striked := False;</p> <p>Color := Referenz auf eine Instanz der Klasse Color, z.B. "Schwarz";</p> <p>ClipBox := 0.1;</p> <p>ClipFont := Referenz auf eine Instanz der Klasse Font, z.B. "Arial-Clip";</p> <p><b>Parameter:</b></p> <p>Name := "Parzellennummer";</p> <p>Txt := "Hello World";</p> <p>Geometry := 600000.0/200000.0;</p> <p>Rotation := 30.0;</p> <p>HAlI := 0;</p> <p>VAlI := 0;</p> <p>Priority := 120;</p>
--	--

#### 4.4.8 Symbolsignatur

<b>Klasse</b>	<pre> CLASS SymbolSign (EXTENDED) =   Name      : NAME;   Symbol    : MANDATORY -&gt; FontSymbol;   Scale     : ES Float;   Color     : -&gt; Color;   Rotation  : ES Angle;   ClipSymbol : -&gt; FontSymbol; PARAMETER   Name      : NAME;   Geometry  : MANDATORY Style COORD2;   Rotation  : Style ANGLE;   Priority   : MANDATORY ES Priority; END SymbolSign; </pre> <p>Kursive Attribute oder Parameter stammen aus gleichnamigen Klasse im Symbologiemodell BasicSymbology .</p>
<b>Beschreibung</b>	<p>Mit der Klasse <code>SymbolSign</code> werden Symbolsignaturen definiert. Die Klasse <code>SymbolSign</code> erweitert die gleichnamige Klasse aus dem SYMBOLOGY MODEL <code>BasicSymbology</code> um effektive Darstellungsattribute, unter anderen auch um Attribute mit Werten aus den vorgängig definierten Signaturbibliotheken wie <code>Color</code>, etc.</p>
<b>Attribut <code>Symbol</code></b>	<p>Definiert das Symbol für die Signatur.</p>
<b>Attribut <code>Scale</code></b>	<p>Definiert die Skalierung des Symbols, respektive der Geometrien des Symbols.</p>
<b>Attribut <code>Color</code></b>	<p>Definiert die Farbe des Symbols. Falls definiert, übersteuert der Wert die Farben der Geometrien des Symbols.</p>
<b>Attribut <code>Rotation</code></b>	<p>Definiert den Standart-Winkel zur Plazierung des Symbols, respektive der Geometrien des Symbols.</p>
<b>Attribut <code>ClipSymbol</code></b>	<p>Definiert das Clippsymbol für das Symbol definiert mit dem Attribut <code>Symbol</code> für die Signatur.</p> <div style="text-align: center;">  </div> <p>Ein Clipping hat den Hintergrund um dieses Clippsymbol auszuschneiden.</p>

<b>Parameter</b> <code>Priority</code>	Definiert die Priorität der Signatur. Eine Priorität mit einem höheren Wert - z.B. 5 - "liegt" über einer Priorität mit einem tieferen Wert - z.B. 4 -.
<b>Beispiel</b>	Definiert wird die Symbolsignatur "Baum". Attribute: <pre> Name           := "Baum"; Symbol         := Referenz auf eine Instanz                 der Klasse FontSymbol, z.B. "Baum"; Scale          := 2.0; Color          := Referenz auf eine Instanz                 der Klasse Color, z.B. "Gruen"; Rotation       := 0.0; ClipSymbol     := Referenz auf eine Instanz                 der Klasse FontSymbol, z.B. "Baum-Clip"; </pre> Parameter: <pre> Name           := "Baum"; Geometry       := 600000.0/200000.0; Rotation       := 30.0; Priority       := 120; </pre>

## 4.4.9 Liniensignatur

<b>Klasse</b>	<pre> CLASS PolylineSign (EXTENDED) =   Name      : NAME;   Style     : LIST OF LineStyle;   Color     : -&gt; Color;   ClipStyle : -&gt; LineStyle;   StartSymbol: -&gt; SymbolSign;   EndSymbol : -&gt; SymbolSign; PARAMETER   Name      : NAME;   Geometry  : MANDATORY Style POLYLINE;   Priority   : MANDATORY ES Priority;   Width     : ES Float; END PolylineSign; </pre> <p>Kursive Attribute oder Parameter stammen aus gleichnamigen Klasse im Symbologiemodell BasicSymbology .</p>
<b>Beschreibung</b>	<p>Mit der Klasse <code>PolylineSign</code> werden Liniensignaturen definiert. Die Klasse <code>PolylineSign</code> erweitert die gleichnamige Klasse aus dem SYMBOLOGY MODEL <code>BasicSymbology</code> um effektive Darstellungsattribute, unter anderen auch um Attribute mit Werten aus den vorgängig definierten Signaturbibliotheken wie <code>Color</code>, etc.</p>
<b>Attribut <code>style</code></b>	<p>Definiert eine geordnete Liste von Linienarten. Die Liste muss im Minimum eine Linienart beinhalten. Für Multilines - z.B. Kanal mit Achse und Wänden - kann die Liste mehrere Linienarten beinhalten.</p>
<b>Attribut <code>color</code></b>	<p>Definiert die Farbe der Liniensignatur. Falls definiert, übersteuert der Wert die Farben der Linienarten.</p>
<b>Attribut <code>clipStyle</code></b>	<p>Definiert die Clipplinienart aus der Linienbibliothek für die Signatur. Die Clipplinienart definiert eine Linienart, die als Maske für die Linienarten im Attribut <code>style</code> dienen.</p> <div data-bbox="735 1529 1193 1568" style="text-align: center;"> </div> <p>Ein Clipping hat den Hintergrund unter der Clipplinienart auszuschneiden.</p>
<b>Attribut <code>startSymbol</code></b>	<p>Definiert ein Symbol am Anfang der Polyline. Der Ursprung für das Symbol ist der erste Punkte der Polyline.</p>
<b>Attribut <code>endSymbol</code></b>	<p>Definiert ein Symbol am Ende der Polyline. Der Ursprung für das Symbol ist der letzte Punkte der Polyline.</p>

<b>Parameter</b> <code>priority</code>	Definiert die Priorität der Signatur. Eine Priorität mit einem höheren Wert - z.B. 5 - "liegt" über einer Priorität mit einem tieferen Wert - z.B. 4 -.
<b>Parameter</b> <code>width</code>	Definiert die Breite der Linienarten, respektive die Skalierung der Geometrien der Linienarten, deren Offset und die Skalierung des Start- und Endsymbols.
<b>Beispiel</b>	<p>Definiert wird die Liniensignatur "Leitung" mit einer Achse und zwei Wänden.</p> <p>Attribute:</p> <pre> Name           := "Leitung"; Style          := Referenz auf eine Liste von Instanzen                 der Klasse LineStyle                 Instanzen sind:                     - eine Achse mit dem Offset=0.0,                     - eine Wand mit einem positiven Offset,                     - eine Achse mit einem negativen Offset; ClipStyle      := Referenz auf eine Instanz                 der Klasse LineStyle; StartSymbol    := Referenz auf eine Instanz                 der Klasse SymbolSign; EndSymbol      := Referenz auf eine Instanz                 der Klasse SymbolSign; </pre> <p>Parameter:</p> <pre> Name           := "Leitung"; Geometry       := 600000.0/200000.0 ... 600100.0/200100.0; Priority        := 120; Width          := 0.1; </pre>

#### 4.4.10 Flächensignatur

<b>Klasse</b>	<pre> CLASS SurfaceSign (EXTENDED) =   Name      : NAME;   FillColor : -&gt; Color;   Border    : -&gt; PolylineSign;   Clip      : (inside, outside);   HatchSymb : -&gt; PolylineSign;   HatchOffset: ES Float; PARAMETER   Name      : NAME;   Geometry  : MANDATORY Style SURFACE;   Priority   : MANDATORY ES Priority;   HatchAng  : ES Angle;   HatchOrg  : ES Coord2; END SurfaceSign; </pre> <p>Kursive Attribute oder Parameter stammen aus gleichnamigen Klasse im Symbologiemodell BasicSymbology.</p>
<b>Beschreibung</b>	<p>Mit der Klasse <code>SurfaceSign</code> werden Flächensignaturen definiert.</p> <p>Die Klasse <code>SurfaceSign</code> erweitert die gleichnamige Klasse aus dem SYMBOLOGY MODEL <code>BasicSymbology</code> um effektive Darstellungsattribute, unter anderen auch um Attribute mit Werten aus den vorgängig definierten Signaturbibliotheken wie <code>Color</code>, etc.</p>
<b>Attribut <code>FillColor</code></b>	<p>Definiert die Füllfarbe der Flächensignatur.</p>
<b>Attribut <code>Border</code></b>	<p>Definiert eine Liniensignatur für die Grenzlinien der Fläche.</p> <p>Falls nicht gesetzt, werden die Grenzlinien nicht dargestellt.</p>
<b>Attribut <code>Clip</code></b>	<p>Definiert den Clippmodus für die Fläche.</p> <p><code>Clip:= #inside</code>, der Hintergrund der Fläche wird geclippt.</p> <p><code>Clip:= #outside</code>, alles was ausserhalb der Fläche liegt wird geclippt.</p> <p>Kann benutzt werden, um über eine Fläche die Ausdehnung eines Planes zu definieren.</p>
<b>Attribut <code>HatchSymb</code></b>	<p>Definiert eine Liniensignatur für das Schraffieren der Fläche.</p> <p>Mit einer Liniensignatur mit einer Instanz der Linienart <code>LineStyle_Pattern</code> wird eine Bemusterung (Pattern) der Fläche erreicht.</p>
<b>Attribut <code>HatchOffset</code></b>	<p>Definiert die Distanz zwischen den Linien der Schraffur.</p>

<b>Parameter</b> <code>Priority</code>	Definiert die Priorität der Signatur. Eine Priorität mit einem höheren Wert - z.B. 5 - "liegt" über einer Priorität mit einem tieferen Wert - z.B. 4 -.
<b>Parameter</b> <code>HatchAngle</code>	Definiert den Winkel für die Linien der Schraffur.
<b>Parameter</b> <code>HatchOrg</code>	Definiert den Ursprung für die Linien der Schraffur. Eine - eventuell auch nicht dargestellte - Linie der Schraffur führt durch diesen Ursprung. Default-Ursprung 0.0/0.0.
<b>Beispiel</b>	<p>Definiert wird die Flächensignatur "Gebäude".</p> <p>Attribute:</p> <pre> Name           := "Gebaeude"; FillColor      := Referenz auf eine Instanz                   der Klasse Color, z.B. "Gruen"; Border         := Referenz auf eine Instanz                   der Klasse PolylineSign; Clip           := #inside; HatchSymb      := Referenz auf eine Instanz                   der Klasse PolylineSign; HatchOffset    := 0.0; </pre> <p>Parameter:</p> <pre> Name           := "Gebaeude"; Geometry       := 600000.0/200000.0 ... 600000.0/200000.0; Priority        := 10; HatchAng       := 45.0; HatchOrg       := 100.0/200.0; </pre>

## 5 Anhänge

### Anhang: Literatur

- [1] INTERLIS Version 2.0 Referencemanual  
Bundesamt für Landestopographie, Eidgenössische Vermessungsdirektion
- [2] Farben in INTERLIS  
Bundesamt für Landestopographie, Eidgenössische Vermessungsdirektion

### Anhang: Impressum

#### Entstehungsgeschichte dieses Dokuments

<i>Version</i>	<i>Datum</i>	<i>Wer</i>	<i>Was</i>
1.0	2000-07-28	TG	Erster ausformulierter Entwurf
1.0	2000-12-13	SK	Satzänderungen, neue Rechtschreibung