esri Suisse

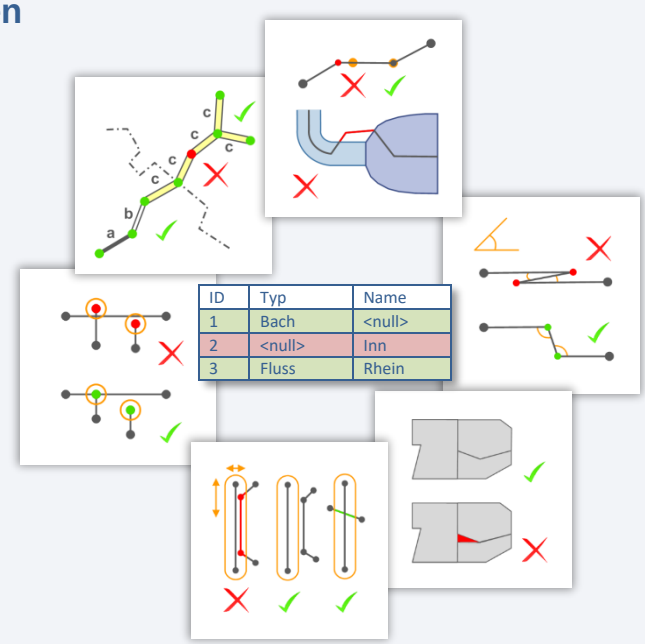
QA-Framework

Test Quick Reference

Das QA-Framework von Esri Schweiz ermöglicht die umfassende und effiziente Qualitätssicherung von Geodaten. Es bietet eine umfangreiche Palette von Werkzeugen für das Identifizieren, Beurteilen und Korrigieren von Fehlern. Zentrales Element sind die Qualitätstests, mit welchen die Daten geprüft werden. Das vorliegende Dokument bietet einen Überblick über diese Tests.

Über 130 Tests in den folgenden Kategorien

- Attribute und Beziehungen
- Geometrie einzelner Features
- Lineare Netzwerke
- Proximität (Abstände zwischen Features)
- Topologische Bedingungen
- M-Werte
- Z-Werte
- 3D-Gebäude (MultiPatch-Features)
- Terrain-Datasets
- Polygonnetzwerke (Grenzlinien und Zentroide)
- Kartographische Symbolisierung
- Grenzprüfung (Randabgleich)
- Schemas von Tabellen/Featureklassen



Merkmale der Tests

- Sehr schnelle Prüfung räumlicher Beziehungen, auch bei sehr großen Datenmengen
- Fehlergeometrien und beteiligte Features werden exakt identifiziert
- Komplexe Attributbedingungen können mit räumlichen Bedingungen kombiniert werden
- Ausnahmen können gespeichert werden
- Features aus unterschiedlichen Datenbanken können gegeneinander geprüft werden
- Für Geodatabases (ArcSDE, Personal und File) und Shapefiles, in ArcGIS 10.0 bis 10.7

Das QA-Framework wird als ProSuite QA Extension für ArcGIS for Desktop angeboten. Außerdem wird es in Individualentwicklungen eingesetzt.

Esri Schweiz AG, Josefstrasse 218, 8005 Zürich, Telefon + 41 58 267 18 00, info@esri.ch, http://esri.ch

Werte entsprechen den Domänen und Subtypen

Findet Einträge in Tabellen und Featureklassen mit Attributwerten, welche die in der Geodatabase definierten Attributregeln (Domänen, Subtypen) verletzen.

Test:*QaGdbConstraintFactory*

Domäne für Feld [Typ]

Code	Beschreibung
100	Pfad
200	4m Straße

Geprüfte Tabelle

	ID	Typ
X	1	300
	2	100

Frei definierbare Attributbedingungen

Findet Einträge in Tabellen und Featureklassen, welche definierte Attributbedingungen nicht erfüllen.

Bedingungen können in Hierarchien gegliedert werden. Dabei können Objektmengen durch Auswahlkriterien schrittweise eingegrenzt werden. Für jede so definierte Objektmenge können zu prüfende Attributbedingungen definiert werden.

Attributbedingungen können auch für Tabellen-Joins geprüft werden.

Attributbedingungen können auch in einer separaten Datenbank-Tabelle definiert sein.

Tests: *QaConstraint, QaDatasetConstraintFactory, QaRelConstraint, QaConstraintListFactory*

```
Objektart=Pfad (Auswahl)
+Breite IS NULL OR
 Breite < 4 (Bedingung)

Objektart=Autobahn (Auswahl)
+Breite > 10 (Bedingung)
+Belag <> ,Natur' (Bedingung)
```

	ID	Objektart	Breite	Belag
X	1	Pfad	6m	Natur
	2	Pfad	3m	Natur
	3	Autobahn	12m	Hart
X	4	Autobahn	12m	Natur

Keine NULL-Werte in Pflichtfeldern

Findet Einträge in Tabellen und Featureklassen, die keine Werte (NULL-Werte) in definierbaren Pflichtfeldern aufweisen.

Test: *QaRequiredFields*

[Typ] ist ein Pflichtfeld.
[Name] ist kein Pflichtfeld.

	ID	Typ	Name
	1	Bach	<null>
X	2	<null>	Inn
	3	Fluss	Rhein

Referentielle Integrität

Findet Einträge in Tabellen und Featureklassen, die Fremdschlüsselwerte enthalten, die sich auf keinen Schlüsselwert in der referenzierten Tabelle beziehen.

Dieser Test unterstützt auch Schlüssel, die aus mehreren Feldern zusammengesetzt sind sowie Tabellen aus unterschiedlichen Datenbanken. Außerdem besteht die Möglichkeit, unerwünschte *bestehende* Referenzen zwischen Untermengen von Tabelleneinträgen als Fehler zu melden.

Test: *QaForeignKey*

Referenzierte Tabelle

ID	Land
1	Schweiz
2	Deutschland

Geprüfte Tabelle

	ID	Stadt	Land FK
X	1	Lissabon	5
	2	Berlin	2
	3	Bern	1

Keine nicht referenzierte Einträge

Findet Einträge in Tabellen und Featureklassen, auf die kein Eintrag in einer oder mehreren referenzierenden Tabellen verweist.

Die Tabellen können in unterschiedlichen Datenbanken vorliegen. Unterstützt werden n:1, 1:1 und n:m-Beziehungen.

Test: *QaUnreferencedRows*

Vermessungslinien

ID	Auftrag-ID
1	10
2	10
3	11

Vermessungspunkte

ID	Auftrag-ID
1	11
2	10

Referenzierte Aufträge

	ID	Nummer	Jahr
	10	AB135324	2012
	11	XY453465	2009
X	12	PP512324	2010

Eindeutige Feldwerte

Findet Einträge in Tabellen und Featureklassen, die auf Grund eines Attributes oder einer Kombination von mehreren Attributen nicht eindeutig sind.

Dieser Test ist auch anwendbar auf mehrere Felder aus verknüpften Tabellen.

Tests: *QaUnique, QaRelUnique*

Der Inhalt der Felder [Region] und [Name] soll zusammen eindeutig sein

	ID	Region	Name
	1	A	X
X	2	A	Y
X	3	A	Y
	4	B	Z
	5	B	X

Reguläre Ausdrücke für Textfelder

Findet Einträge in Tabellen und Featureklassen, bei denen der Inhalt von einem oder mehreren Textfeldern nicht einem definierten regulären Ausdruck (regex) entspricht.

Mit diesem Test lassen sich Textwerte auf frei definierbare Formatregeln prüfen (z.B. strukturierte Schlüssel, Email-Adressen, Telefonnummern, Ausschließen von Sonderzeichen, etc.)

Test: *QaRegularExpression, QaRelRegularExpression*

Keine Leerschläge oder Tabulatoren am Textanfang
Ausdruck: `^[ \t]+`

	ID	Beschreibung
X	1	··Beispieltext
X	2	→ Beispieltext
	3	Beispieltext

Anzahl unterschiedliche Werte innerhalb von Gruppen von Einträgen

Überprüft, ob die Anzahl der unterschiedlichen Resultatwerte eines Ausdrucks innerhalb einer Gruppe von Einträgen in Tabellen und Featureklassen zwischen definierbaren Minimal- und Maximalwerten liegt.

Dieser Test ist auch anwendbar auf Felder aus verknüpften Tabellen. Außerdem können Gruppen über mehrere Tabellen/Featureklassen hinweg ausgewertet werden, mit individuellen Gruppierungs- und Wertkriterien. Als Gruppierungs- und Wertkriterien können einfache Feldnamen oder komplexere SQL-Funktionen verwendet werden.

Anwendungsbeispiele:

- In denormalisierten Tabellenschemas kann geprüft werden, ob pro Wert eines Attributs (Gruppierungskriterium) jeweils ein eindeutiger Wert in einem abhängigen Attribut vorliegt (s. Beispiel rechts).
- Über mehrere Tabellen/Featureklassen hinweg kann geprüft werden, ob innerhalb einer Gruppe nur jeweils Einträge aus der einen oder anderen Tabelle vorhanden sind.

Tests: *QaGroupConstraints, QaRelGroupConstraints*

Informationen zu Flussläufen liegen (denormalisiert) auf den einzelnen Fluss-Linien: pro Fluss-ID muss der Fluss-Name eindeutig sein

	ID	Fluss.Lauf	Fluss.Name
X	1	1	Elbe
X	2	1	Donau
	3	2	Rhein
	4	2	Rhein

Keine Leerschlagzeichen in Textfeldern

Findet Einträge in ausgewählten Textfeldern, die am Beginn oder Ende ein oder mehrere Leerschlagzeichen aufweisen.

Hintergrund: Diese Leerschläge erschweren u.a. Abfragen und sind vor allem am Ende eines Feldinhalts nicht leicht aufzufinden.

Test: *QaTrimmedTextFields*

	ID	Beschreibung
X	1	␣Beschreibung 1
X	2	Beschreibung 2␣
	3	Beschreibung 3

Keine leeren Zeichenketten in Text-Pflichtfeldern

Überprüft, ob Pflichtfelder vom Typ Text leere Zeichenketten enthalten.

Hintergrund: Oracle wandelt leere Zeichenketten automatisch in NULL-Werte um. Dadurch kann der Import aus anderen Datenbanken, welche leere Zeichenketten in Pflichtfeldern enthalten, scheitern (durch Verletzung der NOT NULL-Bedingung in Oracle). Mit Hilfe dieses Tests kann in diesen anderen Datenbanken sichergestellt werden, dass der Export nach Oracle möglich ist.

Test: *QaEmptyNotNullTextFields*

	ID	Beschreibung [NOT NULL]
X	1	""
	2	Beschreibung

Datum ohne Zeitangaben

Findet Einträge in Tabellen und Featureklassen, welche in ausgewählten Datumsattributen zusätzlich zum Datum eine unerwünschte Zeitangabe enthalten.

Hintergrund: Zeitangaben in einem Datumsfeld erschweren die Abfrage aufgrund von Datumswerten, weil dazu entweder Bereichsabfragen (>=, <=) oder SQL-Funktionen für die Umwandlung des Datumswerts nötig sind. In gewissen Fällen sind solche Zeitangaben inhaltlich nicht erwünscht/nötig und können mit diesem Test identifiziert werden.

Test: *QaDateFieldsWithoutTime*

	ID	Datum
X	1	2001-09-13-05-00-30
	2	2001-09-13-00-00-00

Gültige Datumswerte

Findet Einträge in Tabellen und Featureklassen, welche in ausgewählten Attributen einen Datumswert außerhalb eines definierbaren Zeitraums enthalten. Optional kann der Zeitraum durch Datumsangaben relativ zum aktuellen Datum definiert werden.

Hintergrund: neben der spezifischen Prüfung gemäß der Bedeutung eines Datumsfeldes sind damit auch generelle Prüfungen auf technisch begrenzte Datumsbereiche möglich. Zum Beispiel unterstützt SQL Server in DateTime-Feldern nur Werte ab 1.1.1753. In anderen Datenbanken kann mit diesem Test geprüft werden, ob ein späterer Export nach SQL Server/Express aufgrund von früheren Datumswerten (z.B. 1.1.01 verwendet als Spezialwert) scheitern wird.

Test: *QaValidDateValues*

```
minimumDateValue:
  01-01-1990

maximumDateValue:
  31-12-2019
```

	ID	Datum
X	1	1850-09-13
X	2	2020-09-13
	3	2012-09-13

Gültige URLs

Überprüft, ob Attributwerte in einer Tabelle gültige URLs enthalten. Als ungültig gelten sie dann, wenn die Ressource nicht verfügbar ist (z.B. nichtexistierende Datei, Server nicht erreichbar etc.). Unterstützt werden zur Zeit http-URLs oder Filesystem-Verweise (UNC Pfade oder verbundene Netzlaufwerke).

Die URL kann in einem einzelnen Feld gespeichert oder das Resultat eines SQL Ausdrucks sein, der sich auf ein oder mehrere Attribute bezieht und z.B. konstante Pfad-Bestandteile unterstützt.

Test: *QaValidUrls*

	ID	URL
X	1	http://invalid.url.html
	2	http://valid.url.html
	3	\\server\directory\file.txt
	4	file:///T:/directory/file.txt
	5	

Gültige Datentypen

Findet Einträge in Tabellen und Featureklassen, die nicht mit dem Datentyp des Feldes kompatibel sind, z.B. Zeichenketten in einem GUID-Feld.

Test: *QaValue*

	ID	GUID
X	1	{ungültige_guid}
	2	{1AF99E7F-255F-40EA-B711-ECABE73B74B6}

Erwartete Anzahl Einträge in Tabellen oder Featureklassen

Überprüft, ob die Anzahl Einträge in einer Tabelle oder Featureklasse in einem erwarteten Bereich liegt, welcher mit einer definierbaren Unter- und Obergrenze angegeben werden kann, entweder als absolute Zahlen oder relativ zur Anzahl Einträge in einer Referenztabelle. Optional kann die Obergrenze im Test ignoriert werden.

Test: *QaRowCount*

Übereinstimmende Werte in Koordinatenfeldern

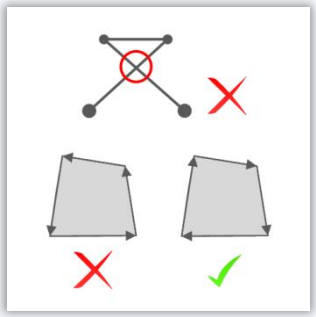
Neu

Findet Punkt-Features mit X, Y und/oder Z-Attributen, deren Werte um mehr als eine definierbare Toleranz von den Feature-Koordinaten abweichen.

Test: *QaValidCoordinateFields*

	ID	Shape	X	Y
	1	Punkt(100,100)	100	100
	2	Punkt(100,100.2)	100	100
X	3	Punkt(100,100)	200	200

Simple Geometrie



Findet Linien-, Polygon-, Multipoint oder Multipatch-Features, deren Geometrie nicht „simple“ ist. Als „simple“ gilt eine Geometrie, wenn die folgenden Bedingungen erfüllt sind:

- Es gibt keine Segmente kürzer als die XY-Toleranz der Featureklasse
- Multipoints: es gibt keine Punktpaare, deren Abstand kleiner ist als die XY-Toleranz der Featureklasse
- Es gibt keine Selbstüberschneidungen
  - Im Fall von Linienfeatures können Selbstüberschneidungen optional erlaubt werden.
- Polygone: Ringe sind korrekt orientiert:
  - äußere Ringe: im Uhrzeigersinn
  - innere Ringe („Löcher“): im Gegenuhrzeigersinn
- Polygone: Ringe sind geschlossen
- Polygone/Polylinien mit mehreren Teilen (Ringe bzw. Pfade): es gibt keine leeren Geometrieteile
- Polygone/Polylinien: Alle Segmente eines Rings bzw. Pfads haben die gleiche Orientierung

Das Vorhandensein von nicht-simplen Geometrien kann dazu führen, dass Geometrieoperationen mit einer Fehlermeldung scheitern. Deshalb sollte dieser Test beispielsweise bei Übernahme externer Daten frühzeitig zur Anwendung kommen.

Test: *QaSimpleGeometry*

Keine Features ohne Geometrie

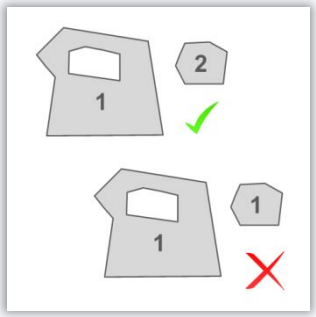
X

ID	SHAPE
0	<null>
1	Polygon

Findet alle Features mit leerer Geometrie.

Test: *QaNonEmptyGeometry*

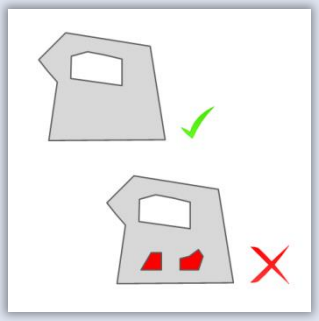
Keine Multipart-Features



Findet Features mit mehreren Geometrieteilen (Multipart-Features). Im Fall von Polygonen werden normalerweise nur mehrere *äußere* Ringe als Fehler gemeldet. Optional werden zusätzlich auch innere Ringe („Löcher“) gemeldet.

Test: *QaMultipart*

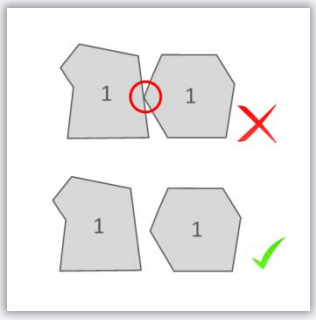
Maximale Anzahl Löcher in Polygonen



Findet Polygon-Features, deren Anzahl innerer Ringe („Löcher“) einen definierten Maximalwert überschreiten.

Test: *QaInteriorRings*

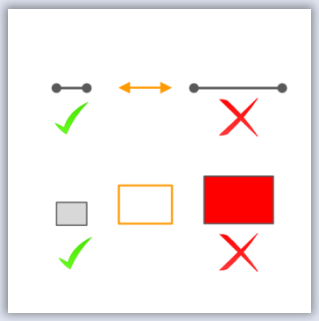
Keine berührende Geometrieteile



Findet Punkte in Multipart-Polygonen oder -Linien, wo sich mehrere Geometrieteile berühren.

Test: *QaNoTouchingParts*

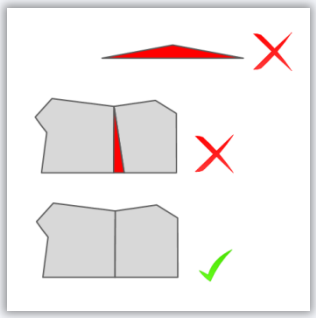
Minimal- und Maximaldimensionen



Findet Linien- und Polygon-Features, deren Länge oder Fläche einen definierbaren Wert über- oder unterschreitet. Bei Multipart-Features können einzelne Geometrieteile optional individuell geprüft werden.

Tests: *QaMaxArea, QaMaxAreaFactory, QaMinArea, QaMinAreaFactory, QaMaxLength, QaMaxLengthFactory, QaMinLength, QaMinLengthFactory*

Keine Sliver-Polygone



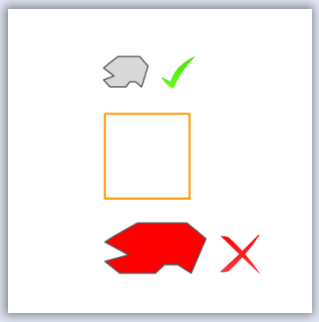
Findet langgestreckte Polygone („Slivers“). Als „Sliver“ wird ein Polygon klassiert, wenn sein Verhältnis von Umfang im Quadrat geteilt durch die Fläche einen bestimmten Wert überschreitet. Die folgende Auflistung zeigt den Wert für (Umfang^2 / Fläche) für Rechtecke unterschiedlicher Seitenverhältnisse:

- Seitenverhältnis 1:1 (Quadrat): 16
- Seitenverhältnis 1:5: 28,8
- Seitenverhältnis 1:10: 48,4
- Seitenverhältnis 1:20: 88,2

Zusätzlich kann optional eine Fläche angegeben werden, welche ebenfalls unterschritten sein muss, damit das Polygon als „Sliver“ gilt.

Test: *QaSliverPolygon*

Maximale Ausdehnung eines Features

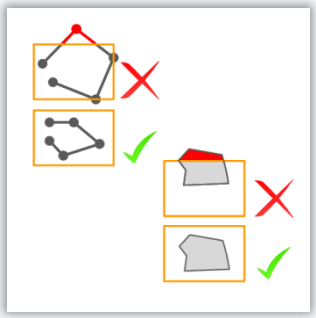


Findet Features, deren maximale Ausdehnung in X- oder Y-Richtung grösser ist als ein definierbarer Maximalwert.

Bei Multipart-Features können einzelne Geometrieteile optional individuell geprüft werden.

Tests: *QaExtent, QaExtentFactory*

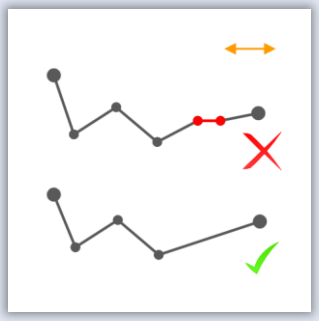
Geometrie innerhalb eines Rechtecks



Findet alle Features, die nicht vollständig in einem spezifizierten Rechteck enthalten sind. Die Grenzen des Rechtecks können mit einem minimalen sowie maximalen X- und Y-Wert definiert werden. Bei teilweise außerhalb des Rechtecks liegenden Features können kann optional das ganze Feature oder nur der außerhalb liegende Teil als Fehler gemeldet werden.

Test: *QaWithinBox*

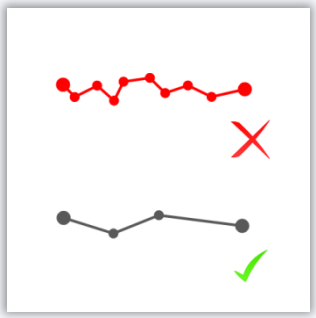
Minimallänge von Segmenten



Findet Segmente von Linien oder Polygon-Features, die kürzer sind als eine definierbare Minimallänge. Bei Features mit Z-Werten kann wahlweise die 2D- oder die 3D-Länge der Segmente geprüft werden.

Test: *QaSegmentLength*

Maximale Anzahl Stützpunkte



Findet Multipoint-, Multipatch-, Linien- oder Polygon-Features, die mehr Stützpunkte aufweisen als ein definierbarer Maximalwert.

Bei Multipart-Features können einzelne Geometrieteile individuell geprüft werden.

Hintergrund: Geometrien mit mehreren Zehn- bis Hunderttausenden von Stützpunkten erhöhen den Rechenaufwand für gewisse Geometrieoperationen massiv und können zu langsamen Antwortzeiten führen. Dieser Test erlaubt, diese Features zu identifizieren. In gewissen Fällen lässt sich die Anzahl Stützpunkte ohne wesentlichen Informationsverlust verringern.

Test: *QaMaxVertexCount*

Minimale Durchschnittssegmentlänge

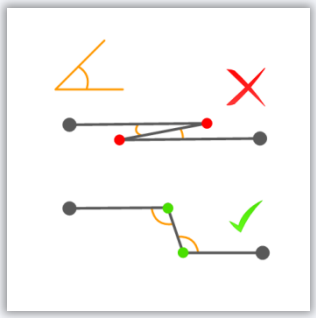
Findet Linien- oder Polygon-Features, deren *durchschnittliche* Segmentlänge einen definierbaren Minimalwert unterschreitet.

Bei Multipart-Features können einzelne Geometrieteile individuell geprüft werden.

Dieser Test ergänzt die Prüfung auf minimale Segmentlänge und maximale Anzahl Stützpunkte, indem Features gefunden werden, die eine zu hohe *Dichte* an Stützpunkten aufweisen, selbst wenn die maximale Anzahl Stützpunkte nicht überschritten oder die minimale Länge eines einzelnen Segments nicht unterschritten wurde.

Tests: *QaMinMeanSegmentLength*

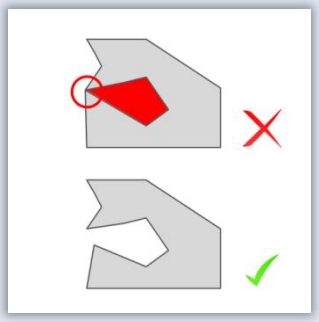
Minimaler Winkel von aufeinander folgenden Segmenten



Findet aufeinander folgende Segmente von Linien oder Polygon-Features, deren Zwischenwinkel kleiner ist als ein definierbarer Minimalwert („Cutbacks“).

Tests: *QaMinSegAngle, QaMinSegAngleFactory*

Keine geschlossenen Schleifen in Flächengrenzen



Findet geschlossene, nach innen gerichtete Schleifen in Grenzen von Polygonen und Multipatch-Ringen („Boundary Loops“).

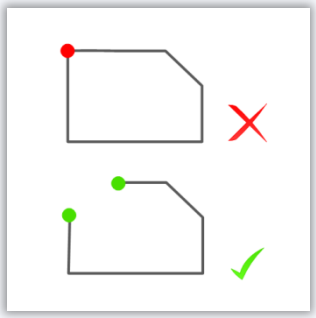
Hintergrund: Diese Schleifen verletzen zwar nicht die Regeln für „simple“ Geometrien, können aber bei Verarbeitung durch andere Systeme dennoch zu Problemen führen. Zu deren Vermeidung kann eine solche Schleife entweder am Berührungspunkt leicht geöffnet, oder durch Verbreiterung der Verbindungsstelle in einen echten inneren Ring („Loch“) umgewandelt werden.

Wahlweise kann die gesamte Fläche, oder nur der Berührungspunkt der Schleife, als Fehler ausgegeben werden. Letzteres erleichtert das Auffinden der zu korrigierenden Stelle im Fall von sehr großen Schleifen.

Optional können Schleifen ignoriert werden, deren Fläche einen definierbare Größe über- oder unterschreitet.

Test: *QaNoBoundaryLoops*

Keine geschlossenen Linien

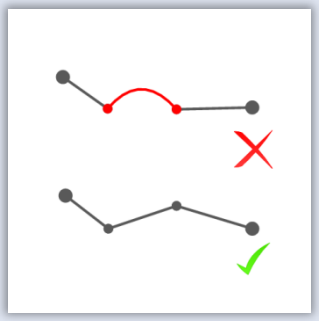


Findet geschlossene Pfade in Linien-Features, d.h. Pfade, bei denen der Start- und der Endpunkt zusammenfallen.

Hintergrund: In geometrischen Netzwerken sind geschlossene Pfade nicht erlaubt. Mit diesem Test können betroffene Features z.B. vor dem Aufbau eines geometrischen Netzwerks identifiziert werden.

Test: *QaNoClosedPaths*

Keine nicht-linearen Segmente

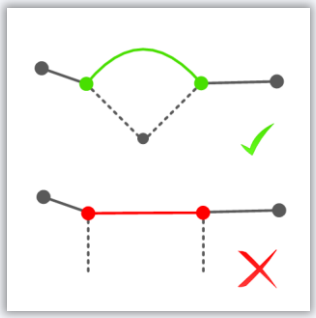


Findet nicht-lineare Geometriesegmente von Linien- und Polygon-Features, wie z.B. Kreisbögen, Ellipsen oder Bézierkurven.

Hintergrund: gewisse Datenformate können keine nicht-linearen Segmente abbilden. Es kommt deshalb vor, dass in gewissen Datenbeständen nicht-lineare Segmente unerwünscht sind, da sie den verlustfreien Export in ein betroffenes Datenformat verunmöglichen.

Test: *QaCurve*

Gültige Kreisbögen



Findet Kreisbogen-Segmente in Linien- und Polygon-Features, welche zu einer geraden Linie degeneriert sind und somit weder einen Mittelpunkt noch einen gültigen Radius aufweisen. Außerdem können Kreisbögen identifiziert werden, deren Segmenthöhe („chord height“) kleiner ist als ein definierbarer Minimalwert.

Hintergrund: diese Art von Kreisbögen können beispielsweise durch den Export in eine Featureklasse mit *größerer* XY-Toleranz des Raumbezugssystems entstehen. Der Grund ist, dass Kreisbögen mit einer Segmenthöhe, welche geringer ist als die XY-Toleranz, in gerade Kreisbögen (mit Radius unendlich und ohne Mittelpunkt) umgewandelt werden. Bei gewissen Arten der Weiterverarbeitung führen diese Kreisbogensegmente anschließend zu Problemen.

Durch die Möglichkeit der Prüfung auf minimale Segmenthöhe kann vor einem solchen Export geprüft werden, ob in der Ziel-Featureklasse degenerierte Kreisbögen entstehen werden.

Test: *QaValidNonLinearSegments*

Bedingungen für Geometrieeigenschaften

Neu

```
($ZMax - $ZMin) < 1000
```

	ID	Shape	*$ZMin*	*$ZMax*
	1	Polygon	*200*	*210*
	2	Polygon	*150*	*170*
X	3	Polygon	*200*	*1300*

Findet Features, für welche eine definierte Bedingung für Geometrieeigenschaften nicht erfüllt ist. Die Bedingung kann die folgenden Geometrieeigenschaften verwenden, und kann optional auf einzelne Teile von Multipart-Geometrien angewendet werden.

- **`$Area`**: die Fläche der Geometrie
- **`$Length`**: die Länge der Geometrie
- **`$VertexCount`**: Anzahl Stützpunkte
- **`$SliverRatio`**: das Verhältnis von Perimeter^2 / Fläche, ein Mass für die Streckung.
- **`$Dimension`**: die Dimension der Geometrie (0, 1, 2)
- **`$EllipticArcCount`**: die Anzahl von elliptischen Bögen
- **`$CircularArcCount`**: die Anzahl der Kreisbögen
- **`$BezierCount`**: die Anzahl der Bézier-Segmente
- **`$LinearSegmentCount`**: die Anzahl linearer Segmente
- **`$NonLinearSegmentCount`**: die Anzahl nicht-linearer Segmente (Kreisbögen etc.)
- **`$SegmentCount`**: die Gesamtzahl der Segmente (linear + nicht-linear)
- **`$IsClosed`**: gibt für Polylinien und Polygone an, ob die Geometrie geschlossen ist.
- **`$XMin`**: minimale X-Koordinate/Rechtswert
- **`$YMin`**: minimale Y-Koordinate/Hochwert
- **`$XMax`**: maximale X-Koordinate/Rechtswert
- **`$YMax`**: maximale Y-Koordinate/Hochwert
- **`$ZMin`**: minimaler Z-Wert
- **`$ZMax`**: maximaler Z-Wert
- **`$MMin`**: minimaler M-Wert
- **`$MMax`**: maximaler M-Wert
- **`$UndefinedMValueCount`**: Anzahl Stützpunkte ohne definierte M-Werte

Zu verwenden, wenn die Bedingung auf die *gesamte* Geometrie angewendet wird:

- **`$PartCount`**: die Anzahl der Geometrieteile
- **`$IsMultipart`**: gibt an, ob die Geometrie aus mehreren Teilen besteht (Ringe, Pfade)
- **`$ExteriorRingCount`**: die Anzahl äusserer Ringe des Polygons
- **`$InteriorRingCount`**: die Anzahl innerer Ringe ("Löcher") des Polygons

Zu verwenden für einzelne Geometrieteile:

- **`$IsExteriorRing`**: gibt an, ob es sich um einen äusseren Ring handelt

- **`$IsInteriorRing`**: gibt an, ob es sich um einen inneren Ring handelt

Zu verwenden mit MultiPatch-Geometrien:

- **`$RingCount`**: die Anzahl der Ringe im MultiPatch
- **`$TriangleFanCount`**: die Anzahl der Dreiecksfächer im MultiPatch
- **`$TriangleStripCount`**: die Anzahl der Dreiecksstreifen im MultiPatch
- **`$TrianglesPatchCount`**: die Anzahl der Dreiecks-Patches im MultiPatch

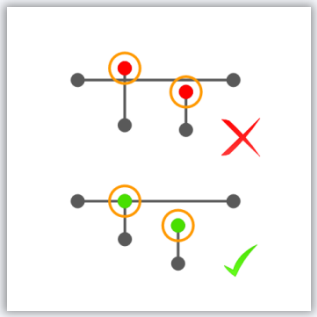
Diese Eigenschaften können in einem SQL WHERE-Ausdruck verwendet werden. Wenn der Ausdruck für ein Feature *nicht* erfüllt ist, dann wird ein Mangel gemeldet.

Beispiele:

```
$SliverRatio < 50 OR $Area > 10
$Area < 5 AND $VertexCount < 10
$Length > 10 OR $PartCount > 1
NOT ($IsClosed AND $EllipticArcCount = 1 AND $SegmentCount = 1)
```

Test: *QaGeometryConstraint*

Keine Linienüber- und unterstände



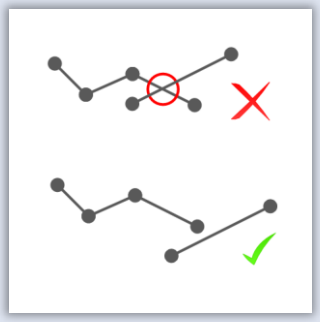
Findet Punkt-Features oder Linienendpunkte, welche nahe bei anderen Linien oder Polygonumrandungen liegen, aber nicht genau mit ihnen zusammenfallen. Mit diesem Test lassen sich somit Linienüber- oder Unterstände finden.

(End-)Punkte, welche weiter als eine definierbare Distanz von der nächsten Linie bzw. Polygonumrandung entfernt liegen, werden nicht als Fehler gemeldet. Im Fall von Featureklassen mit Z-Werten kann diese Distanz wahlweise als 2D- oder 3D-Distanz berechnet werden. Dies erlaubt die kontrollierte Anwendung dieser Prüfung auch bei mehrstufigen Verkehrskreuzungen.

Für das Zusammenfallen eines (End-)Punkts mit einer anderen Linie oder Polygongrenze gilt die XY-Toleranz des Raumbezugssystems.

Test: *QaNodeLineCoincidence*

Keine ungültigen Linienschnittpunkte



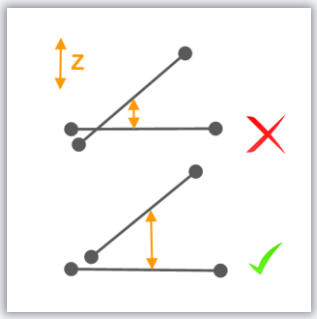
Findet Linien-Features, die sich unerlaubterweise mit einer anderen Linie kreuzen. Es können sowohl Linienüberschneidungen in einem Punkt als auch lineare Überlappungen als Fehler gemeldet werden.

Beim Aufeinandertreffen von Linienendpunkten mit dem Innenbereich einer anderen Linie kann spezifiziert werden, ob die Überschneidung an einer beliebigen Stelle, exakt auf einem Stützpunkt oder gar nicht erlaubt sein soll. Im letzten Fall dürften sich somit nur Linienendpunkte schneiden.

Ausnahmen können über eine SQL-Bedingung definiert werden, mit welchem Attributwerte der am Schnittpunkt beteiligten Features verglichen werden können. Wird die Bedingung erfüllt, gilt die Überschneidung als erlaubt.

Test: *QaLineIntersect*

Minimaler Höhenunterschied bei Linienüberschneidungen



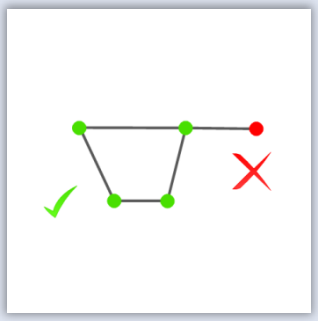
Findet Überschneidungen, wo der Unterschied der Z-Werte der involvierten Linien-Features kleiner ist als ein definierbarer Minimalwert. Optional kann auch eine Obergrenze für den Z-Unterschied festgelegt werden.

Zusätzlich kann eine SQL-Bedingung definiert werden, mit welchem die Attributwerte der am Schnittpunkt beteiligten Features verglichen werden können. Das gemäß Z-Wert am Schnittpunkt höher gelegene Feature kann dabei mit dem Alias „U“ angesprochen werden, das tiefer gelegene Feature mit „L“. Mit dem Ausdruck `U.STUFE > L.STUFE` kann beispielsweise geprüft werden, dass die überquerende Linie einer höheren „Stufe“ zugeordnet ist als das unterquerende Feature.

Wird der minimale Z-Unterschied auf 0 gesetzt, wird ausschließlich die attributive Bedingung getestet.

Test: *QaLineIntersectZ*

Keine „Dangles“ (nicht verknüpfte Linienendpunkte)

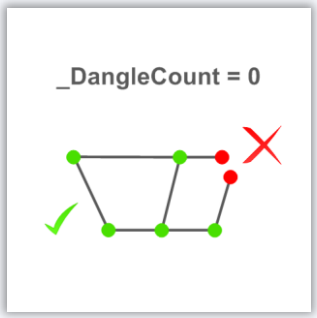


Findet Endpunkte von Linien, die nicht mit anderen Linienendpunkten aus einer Liste von Featureklassen zusammenfallen („Dangles“).

Hinweis: Bei diesem Test werden *alle* Verknüpfungen *nach* Anwendung der Filterbedingungen berücksichtigt. Spezifischere, attributgesteuerte Bedingungen können aber mit *QaDangleCount* und *QaLineConnection* definiert werden.

Test: *QaDangleFactory*

Gültige Anzahl von „Dangles“ (nicht verknüpfte Linienendpunkte)



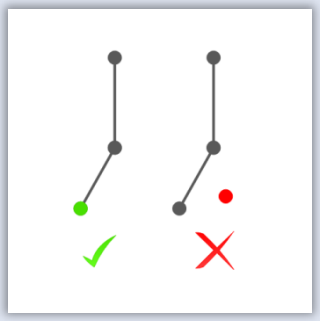
Findet Linien-Features, die eine ungültige Anzahl von nicht verknüpften Endpunkten aufweisen („Dangles“). Die pro Feature erlaubte Anzahl „Dangles“ kann dabei mit einem SQL-Ausdruck bestimmt werden. In diesem Ausdruck kann die effektive Anzahl „Dangles“ als berechnetes Attribut `_DangleCount` abgefragt werden. Ein Fehler wird gemeldet, wenn der SQL-Ausdruck für ein Feature nicht erfüllt ist.

Beispiel für SQL-Ausdruck: Features, welche mittels des Attributs `is_dead_end` als Sackgassen gekennzeichnet sind, müssen genau ein unverknüpftes Ende haben, alle anderen müssen an beiden Enden verknüpft sein.

```
_DangleCount == iif( is_dead_end == 'yes', 1, 0 )
```

Test: *QaDangleCount*

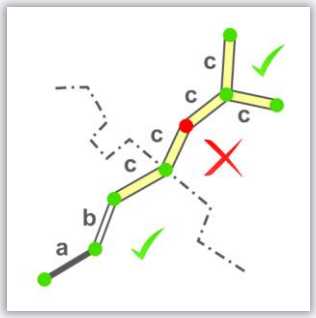
Keine verwaisten, nicht mit Linien verknüpften Punkte



Findet Punkt-Features, die nicht mit Start- oder Endpunkten von Linien-Features zusammenfallen. Umgekehrt können auch Linien-Endpunkte identifiziert werden, die nicht mit einem Punkt-Feature zusammenfallen. Letzteres kann zur Prüfung des exakten Zusammenfallens von Junction-Features mit Linienendpunkten in geometrischen Netzwerken verwendet werden.

Test: *QaOrphanNode*

Keine Pseudoknoten



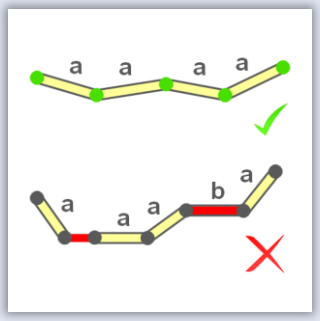
Findet Knoten zwischen genau zwei Linien, welche bei allen relevanten Attributen die gleichen Attributwerte aufweisen („Pseudoknoten“), und die nicht durch ihre Lage zu weiteren Features (Punkte, Grenzlinien) begründet sind.

Standardmässig werden alle editierbaren Felder einer Featureklasse als relevant betrachtet. Optional können spezifische Attribute vom Test ausgeschlossen werden: Unterschiede nur bei diesen Attributen führen dann nicht dazu, dass ein Knoten als gültig erachtet wird.

Verknüpfungsstellen, wo ausnahmsweise ein Pseudoknoten erlaubt ist, können in zusätzlichen Punkt-, Linien- oder Polygon-Featureklassen definiert werden. So können beispielsweise bestimmte Knotentypen oder Aufteilungen auf Grenzlinien erlaubt werden.

Test: *QaFactoryPseudoNodes*

Keine Lücken in gruppierten Linienverläufen



Findet Lücken in Gruppen von Linien aus einer oder mehreren Featureklassen, die eine zusammenhängende Struktur bilden sollen (z.B. Flüsse, Straßenrouten).

Die Gruppierung erfolgt über ein oder mehrere Attribute oder einer verknüpften Tabelle. Dabei werden alle Features mit denselben Werten für diese(s) Attribut(e) derselben Gruppe zugeordnet. Ein Feature kann mehreren Gruppen zugeordnet sein, welche unter Verwendung eines konfigurierbaren Trennzeichens in demselben Textfeld gespeichert sind. Dadurch ist z.B. der Verlauf mehrerer Routen auf denselben Straßenfeatures unterstützt.

Falls eine Gruppe aus nicht zusammenhängenden Features besteht, wird die kürzeste Linie zwischen nicht verbundenen Teilen als Fehler gemeldet.

Optional können spezielle, übergeordnete Strukturen wie Zyklen, Verzweigungen etc. als Fehler gemeldet werden.

Test: *QaGroupConnected, QaRelGroupConnected*

Lineare Netzwerke

Bedingungen für lineare Netzwerke

Erlaubte Linien- und Punktverbindungen

Prüft, ob an Verbindungsstellen zwischen Linien- und Punktfeatures definierte Verbindungsregeln erfüllt sind. Es kann eine Liste von Regeln definiert werden, welche erlaubte Feature-Kombinationen an einer Verbindungsstelle definieren. Trifft keine der Regeln auf eine Verbindungsstelle zu, wird für diese Verbindungsstelle ein Fehler gemeldet.

Regeln beinhalten Auswahlkriterien für Features, sowie optional Bedingungen für die Anzahl von involvierten Features der jeweiligen Featureklassen (wiederum auch für Unter-Selektionen). Im Fall von Linienfeatures kann dabei auch auf die Richtung der Linie am Verbindungspunkt (eingehend, abgehend) Bezug genommen werden.

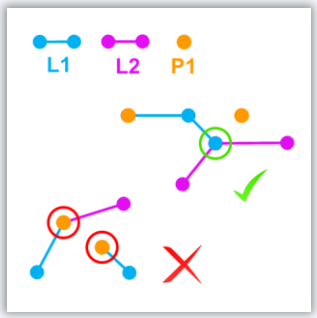
Als Verbindungsstellen gelten alle Linienendpunkte und Punkte aus der Liste der angegebenen Featureklassen. Für jede dieser Verbindungsstellen werden allfällige weitere an dieser Stelle anstoßende Features identifiziert. Auf diese Menge der Features wird anschließend die Prüfung der Verbindungsregeln angewandt.

Die Regeln werden somit auch auf nicht verknüpfte Linienendpunkte, nicht verknüpfte Punkte, und Linienverknüpfungen ohne verknüpfte Punktobjekte angewandt. Anhand der Definition von Bedingungen zur Anzahl verknüpfter Objekte kann an solchen Stellen beispielsweise erkannt werden, dass eine erforderliche Verknüpfung mit anderen Features fehlt.

Test: *QaLineConnection*

Feature-klasse	P1	L1	L2
Regel-gruppe 1	True	True	False
Regel-gruppe 2	False	True	True

1. Verknüpfungen von P1 und L1 sind erlaubt.
   Punkte P1 dürfen nicht mit Linien aus L2 verknüpft sein.
2. Verknüpfungen von L1 und L2 sind erlaubt, sofern kein Punkt P1 an derselben Stelle vorhanden ist.

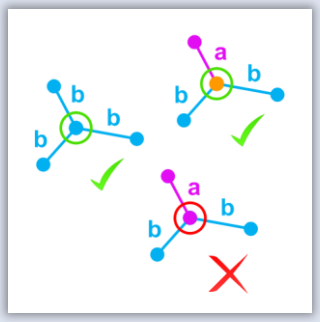


Konsistente Attributwertverteilungen bei Linienverbindungsstellen

Überprüft Bedingungen für die einheitliche Ausprägung von relevanten Attributwerten an Verbindungsstellen von verknüpften Linien und Punkt-Features.

Für die Linien-Features wird definiert, ob alle Werte eines relevanten Attributs identisch sein müssen, oder ob sie unterschiedlich sein *dürfen* oder *müssen*. Für Punkt-Features kann außerdem gefordert werden, dass ein gültiger Punkt an der Linienverbindungsstelle existieren muss, und optional, dass dieser einen Attributwert aufweisen muss, der dem häufigsten/irgendeinem Wert eines Attributs der verknüpften Linien entsprechen muss.

Test: *QaLineConnectionFieldValues*



Konsistente Fließrichtung an Verbindungspunkten von Linien-Features

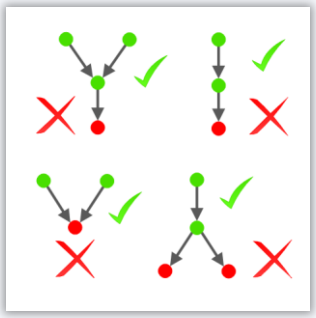
Prüft die Fließlogik in einem Liniennetzwerk. Als Fehler gemeldet werden alle Linienendpunkte, die nicht mit exakt einem Startpunkt eines anderen Linienfeatures verknüpft sind.

Im Normalfall wird angenommen, dass die Fließrichtung der Digitalisierrichtung entspricht. Optional kann ein SQL-Ausdruck angegeben werden, mit welchem Features identifiziert werden, bei welchen die Fließrichtung entgegen der Digitalisierrichtung verläuft.

Zurzeit werden keine übergeordneten Strukturen berücksichtigt, wie z.B. Verzweigungen, die weiter flussabwärts wieder zusammenfließen.

Die beteiligten Features können, müssen aber nicht in einem geometrischen Netzwerk enthalten sein. Eine allfällige, darin berechnete Fließrichtung wird nicht berücksichtigt.

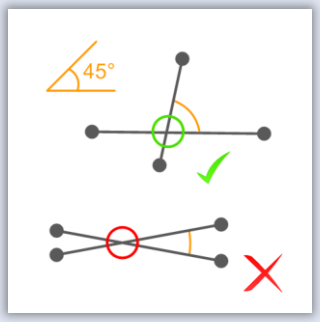
Test: *QaFlowLogic*



Minimalwinkel bei Linienüberschneidungen

Findet Schnittstellen zwischen Linienfeatures, bei denen der Winkel zwischen zwei sich schneidenden Linien kleiner ist als ein definierbarer Minimalwert.

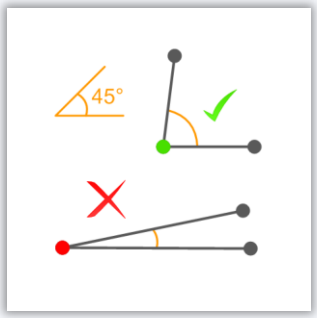
Test: *QaLineIntersectAngle, QaLineIntersectAngleFactory*



Keine spitze Winkel zwischen Linien

Findet Verbindungsstellen, bei denen der Winkel zwischen zwei aufeinandertreffenden Linien kleiner ist als ein definierbarer Minimalwert.

Test: *QaMinAngle, QaMinNodeAngleFactory*



Keine ungültigen Netzwerkelemente

Findet verschiedene technische Probleme, die innerhalb von geometrischen Netzwerken auftreten können.

Der Test prüft, ob Netzwerk-Features gültige zugeordnete Element-ID's des logischen Netzwerks haben. Im Fall von komplexen Kantenfeatures wird außerdem geprüft, ob für jedes enthaltende Teil-Element die Geometrie ausgelesen werden kann, und ob diese gültig (insbesondere nicht leer) ist.

Test: *QaGdbNetworkElements*

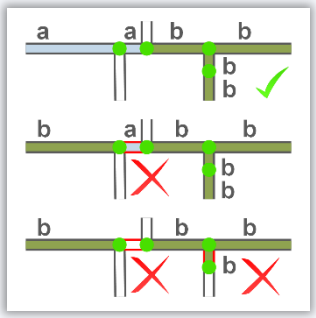
Bedingungen für Liniengruppen

Neu

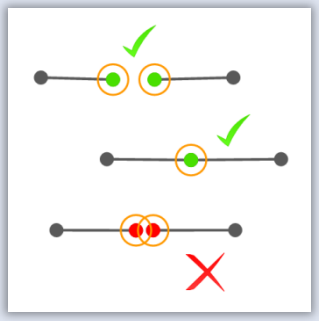
Findet typische Attributierungsfehler in linearen Netzwerken, bei welchen einzelne Features, welche zu einer größeren Struktur gehören sollten (z.B. eine Straßenroute oder ein Gewässerlauf), fehlende/inkorrekte Attributwerte aufweisen. Um diese Situationen zu erkennen, können Bedingungen für zusammenhängende Gruppen von Linienfeatures definiert werden. Diese Bedingungen können die Gesamtlänge einer Gruppe, die lineare Verlaufslänge bis zu Endpunkten der Gruppe, sowie die Länge von Lücken zu anderen Gruppen entlang des darunterliegenden Netzwerks betreffen.

Die Gruppierung erfolgt über ein oder mehrere Attribute oder einer verknüpften Tabelle. Dabei werden alle Features mit denselben Werten für diese(s) Attribut(e) derselben Gruppe zugeordnet. Ein Feature kann mehreren Gruppen zugeordnet sein, welche unter Verwendung eines konfigurierbaren Trennzeichens in demselben Textfeld gespeichert sind. Dadurch ist z.B. der Verlauf mehrerer Routen auf denselben Straßenfeatures unterstützt.

Test: *QaLineGroupConstraints, QaRelLineGroupConstraints*



Punkte und Linienendpunkte fallen zusammen oder halten einen Minimalabstand ein

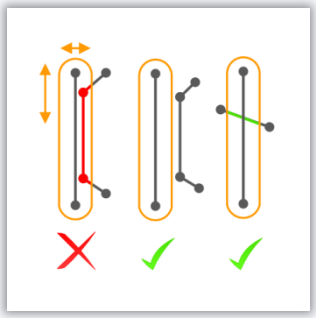


Findet Punkt-Features oder Linienendpunkte, welche näher als eine definierbare Minimaldistanz beieinanderliegen, aber nicht exakt zusammenfallen. Bei exakt zusammenfallenden Knoten von Features mit Z-Koordinaten kann zusätzlich geprüft werden, ob die Z-Werte nicht zu weit auseinanderliegen.

Um zu beurteilen, ob zwei Punkte als zusammenfallend betrachtet werden, wird im Normalfall die XY-Toleranz des Raumbezugssystems verwendet. Optional kann eine davon abweichende Toleranz definiert werden. Bei einer negativen Toleranz gilt auch das Zusammenfallen von Punkten als Fehler. Somit kann mit diesem Test auch auf generelle Einhaltung eines Minimalabstandes zwischen Punkten und/oder Linienendpunkten geprüft werden.

Tests: *QaMinNodeDistance, QaMinNodeDistanceFactory*

Linienstücke halten den Minimalabstand zu einer anderen Linie ein



Findet Linienstücke (Abschnitte auf Linien oder Polygonumgrenzungen), die einen Minimalabstand zu einer anderen Linie/Polygonumgrenzung über eine definierbare Minimallänge unterschreiten.

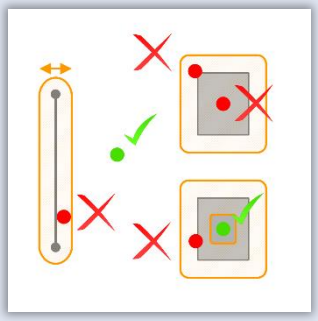
Dieser Test kann z.B. verwendet werden, um doppelt erfasste und somit ähnliche Features zu erkennen. Anhand der definierbaren Minimallänge kann festgelegt werden, über welche Strecke zwei Features einen ähnlichen Verlauf haben müssen, damit sie als Fehler gemeldet werden. Die Minimallänge dient also dazu, längere, nahe beieinanderliegende parallele Linienverläufe von „normalen“ Annäherungen im Rahmen von Linienverbindungen oder Kreuzungen zu unterscheiden.

Mit einer Minimallänge von 0 kann *jedes* Unterschreiten des Minimalabstands als Fehler gemeldet werden.

Test: *QaNotNear*

Punkte halten Minimalabstand zu Linien und Polygongrenzen ein

Neu



Findet Punkt-Features, welche näher als eine definierte Distanz bei Linien- oder Polygon-Features liegen.

Für jede Linien- und Polygon-Featureklasse kann spezifiziert werden, welcher Teil der Geometrie in die Prüfung einfließen soll. Es kann unterschieden werden zwischen der Gesamtgeometrie (Default), der Geometriegrenze, den Stützpunkten, jeweils beiden Linienendpunkten, und entweder nur dem Start- oder Endpunkt von Linien.

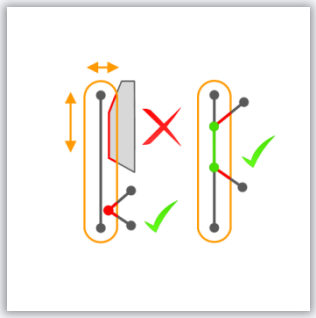
Optional können Punkte erlaubt werden, welche exakt auf der Referenz-Geometrie liegen.

Die minimal einzuhaltende Distanz kann entweder als Konstante angegeben werden, oder aufgrund von Attributwerten der beteiligten Features berechnet werden.

Ausnahmen können anhand von SQL-Ausdrücken definiert werden, in welchen die Attribute von zwei nahe beieinanderliegenden Features verglichen werden können. In diesen Ausdrücken kann das Punkt-Feature mit „G1“ und das Referenz-Feature mit „G2“ angesprochen werden. Ist dieser Ausdruck für die zwei Features erfüllt, dann wird kein Fehler gemeldet. Ein solcher Ausdruck kann entweder für alle Referenz-Featureklassen gelten, oder für jede Referenz-Featureklasse kann ein individueller Ausdruck angegeben werden.

Test: *QaPointNotNear*

Linienstücke halten die Minimaldistanz zu einer anderen Linie ein oder fallen zusammen



Findet Linienstücke (Abschnitte auf Linien oder Polygonumgrenzungen), die einen Minimalabstand zu einer anderen Linie über eine definierbare Minimallänge unterschreiten, ohne mit der anderen Linie zusammenzufallen.

Für die Minimallänge können unterschiedliche Werte definiert werden, einerseits für den Fall, dass eine (Grenz-)Linie im Verlauf innerhalb des Annäherungsbereichs mit der Nachbarlinie zusammenfallen wird, und andererseits für den Fall, dass sie diesen Annäherungsbereich wieder verlässt, ohne je mit der Nachbarlinie zusammenzufallen.

Es kann eine Toleranz für das Zusammenfallen der Linienverläufe angegeben werden. Wenn diese auf 0 gesetzt wird, müssen die Linienverläufe exakt aufeinanderliegen, damit sie als zusammenfallend gelten. Somit können auch kleinste Abweichungen unterhalb der XY-Toleranz des Raumbezugssystems erkannt werden.

Proximität

Bedingungen für Abstände zwischen Features

Mit diesem Test können kleine Verschiebungen zwischen Linien und/oder Polygonen, deren Grenzen zusammenfallen sollten, erkannt werden. Damit lassen sich neben Erfassungsfehlern auch topologische Folgeprobleme von Prozessierungsschritten (z.B. geodätische Transformationen) erkennen.

Dieser Test wird oft zusammen mit *QaNoGaps* und *QaInteriorIntersectsSelf* verwendet, um ein vollständiges Bild der Annäherungen/Lücken/Überlappungen zwischen Polygonen zu erhalten.

Dieser Test existiert in zwei Varianten, einmal für den Vergleich zwischen Features aus zwei *unterschiedlichen* Mengen von Featureklassen, und einmal für den Vergleich zwischen allen Features innerhalb *einer* Menge von Featureklassen.

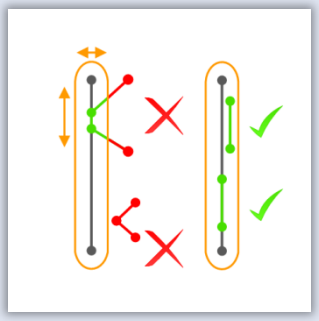
Test: *QaPartCoincidenceOther, QaPartCoincidenceSelf*

Linienverläufe liegen vollständig in der Nähe von Referenzlinien

Findet alle Abschnitte von Linien- oder Polygon-Features, die weiter als eine definierbare Maximaldistanz von einem anderen Linien- oder Polygon-Feature entfernt liegen. Bei Polygon-Features wird dabei jeweils die Umgrenzungslinie der Polygone verwendet.

Bei Features mit Z-Werten kann die Toleranz optional als 2D- oder 3D-Distanz gemessen werden.

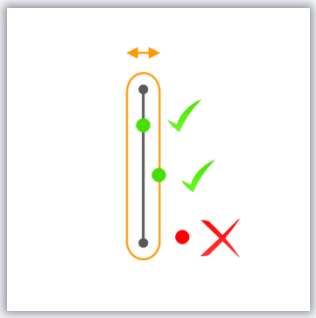
Test: *QaFullCoincidence*



Punkte liegen auf Linien

Findet Punkt-Features, die weiter entfernt von einem Linien-Feature oder einer Polygonumgrenzung liegen als ein erlaubter Maximalabstand.

Test: *QaPointOnLine*

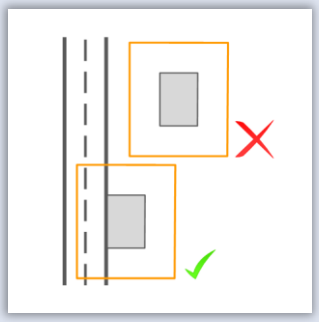


Erwartetes Feature in der Nähe

Findet Features, für welche ein erwartetes Nachbarsfeature innerhalb einer definierten Distanz fehlt. Beispiele sind Bushaltestellen in der Nähe von Straßen, erforderliche Beschriftungsobjekte o.ä.

Optional kann ein SQL-Ausdruck definiert werden, womit die Attribute zweier sich nahe liegenden Features verglichen werden. In diesem Ausdruck können die involvierten Features mit „G1“ und „G2“ angesprochen werden. Ist dieser Ausdruck für die zwei Features nicht erfüllt, dann gilt die räumliche Nachbarschaft als nicht relevant. In diesem Fall wird ein Fehler gemeldet, sofern nicht eine weitere, relevante Nachbarschaft zu einem anderen Feature existiert.

Test: *QaMustBeNearOther*



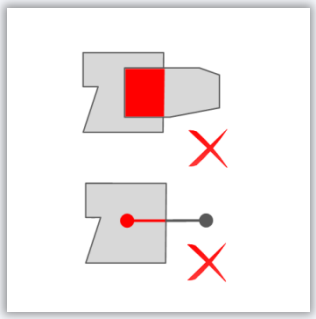
Keine Verletzung von Regeln einer Geodatabase-Topologie

Prüft alle Validierungs-Regeln einer definierten Geodatabase-Topologie, und integriert die identifizierten Topologie-Fehler in die Fehlerverwaltung des QA-Frameworks.

Falls der Prüfprozess eine Bearbeitung der Daten erlaubt, dann wird im Rahmen der Ausführung dieses Tests die Topologie im zu prüfenden Bereich validiert. Dabei können die geprüften Daten im Rahmen des „cracking“ und „clustering“ der Topologie-Featureklassen modifiziert werden. Die dabei identifizierten Fehler werden gemeldet. Falls eine Bearbeitung nicht erlaubt ist, wird die Topologie-Validierung nicht durchgeführt, stattdessen werden lediglich die bereits vorgängig identifizierten Topologie-Fehler weitergemeldet, bzw. eventuell vorhandene Flächen mit nicht geprüften Änderungen („Dirty Areas“) werden gesamthaft als Fehler gemeldet.

Test: *QaGdbTopology*

Keine innere Überschneidung



Findet Features, deren Innenbereich sich mit dem Innenbereich eines anderen Features überschneiden („Überlappungen“ im umgangssprachlichen Sinn).

Optional kann ein SQL-Ausdruck definiert werden, in welchem die Attribute der an einer Überschneidung beteiligten Features verglichen werden können. In diesem Ausdruck können die beteiligten Features mit „G1“ und „G2“ angesprochen werden. Ist dieser Ausdruck für zwei sich überschneidende Features erfüllt, dann gilt die Überschneidung als erlaubt und es wird kein Fehler gemeldet.

Ein weiterer SQL-Ausdruck kann angegeben werden um gewisse Überschneidungen aufgrund von Eigenschaften der Schnittgeometrie zu ignorieren (z.B. Schnittflächen kleiner als ein definierter Schwellenwert). S. die Beschreibung zu „“ (Geometrie-Tests) für eine Liste der unterstützten Geometrieeigenschaften.

Beispiel: der Ausdruck

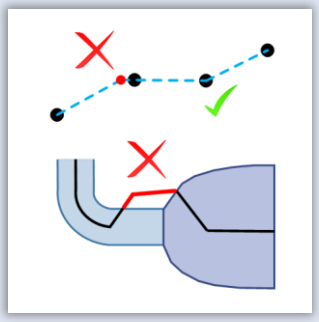
```
$Area < 0.1 AND $SliverRatio > 100
```

ignoriert kleine und langgestreckte Schnittflächen.

Dieser Test existiert in zwei Varianten, einmal für den Vergleich zwischen Features aus zwei *unterschiedlichen* Mengen von Featureklassen, und einmal für den Vergleich zwischen allen Features innerhalb *einer* Menge von Featureklassen.

Test: *QaInteriorIntersectsOther, QaInteriorIntersectsSelf*

Vollständige Abdeckung



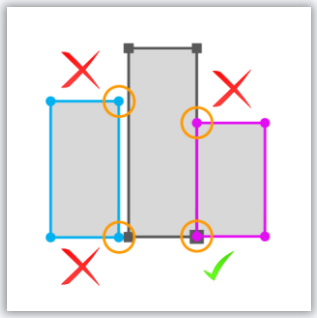
Findet Features, die nicht vollständig von anderen Features abgedeckt werden. Die außerhalb der anderen Features gelegenen Geometrieteile werden als Fehler gemeldet. Dabei kann für jede beteiligte Featureklasse spezifiziert werden, welcher Teil der Geometrie in die Prüfung einfließen soll. Es kann unterschieden werden zwischen der Gesamtgeometrie (Default), der Geometriegrenze, den Stützpunkten, jeweils beiden Linienendpunkten, und entweder nur dem Start- oder Endpunkt von Linien.

Für den Fall, dass ein Feature nicht vollständig abgedeckt sein muss, kann ein maximal erlaubter Prozentsatz für den nicht abgedeckten Teil definiert werden. Ist der Anteil der nicht abgedeckten Fläche bzw. Länge eines Features grösser als dieser Wert, wird ein Fehler gemeldet. Zusätzlich kann eine Distanz um die abdeckenden Features angegeben werden, innerhalb welcher die abzudeckenden Features liegen müssen.

Optional kann ein SQL-Ausdruck definiert werden, in welchem die Attribute eines abzudeckenden Features mit jenen eines potentiell abdeckenden Features verglichen werden können. In diesem Ausdruck kann das potentiell abdeckende Feature mit „G1“ und das untersuchte Feature mit „G2“ angesprochen werden. Falls ein solcher Ausdruck definiert ist, und er für zwei Features nicht erfüllt ist, dann wird die entsprechende Abdeckung ignoriert, was zur Meldung eines Fehlers führen kann (sofern nicht ein anderes Feature mit gültiger Abdeckung vorhanden ist).

Test: *QaIsCoveredByOther*

Übereinstimmung von Stützpunkten und Kanten



Findet Stützpunkte, die nicht mit einem Stützpunkt und/oder einer Kante eines benachbarten Features zusammenfallen. Diese Prüfung kann auch innerhalb der Geometrieteile eines Features erfolgen (beispielsweise für Multipatches).

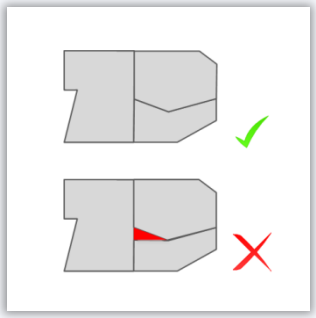
Optional kann definiert werden, ob ein Fehler gemeldet wird, wenn ein zusammenfallender Stützpunkt auf der benachbarten Kante fehlt, auch wenn diese Kante exakt auf dem geprüften Stützpunkt verläuft.

Hintergrund: Nicht exakt aufeinander gesnappte Stützpunkte, bzw. fehlende Stützpunkte auf ansonsten zusammenfallenden Linienverläufen führen zu kleinsten Klaffungen bzw. Überlappungen zwischen Features. Mit diesem Test können die verursachenden Stützpunkt-Abweichungen exakt identifiziert werden.

Dieser Test existiert in drei Varianten: für den Vergleich zwischen Features aus zwei *unterschiedlichen* Mengen von Featureklassen, für den Vergleich zwischen allen Features innerhalb *einer* Menge von Featureklassen, sowie für die Prüfung nur innerhalb *eines* Features (v.a. anwendbar für MultiPatch-Features).

Test: *QaVertexCoincidenceOther, QaVertexCoincidenceSelf, QaVertexCoincidence*

Keine Lücken zwischen Polygonen



Findet Lücken zwischen Polygonen. Optional kann die Suche eingeschränkt werden auf kleine und/oder langgestreckte Lücken. Dazu können eine Maximalgröße und ein maximaler Wert für das Verhältnis von Umfang im Quadrat geteilt durch die Fläche angegeben werden. Die folgende Auflistung zeigt den Wert für (Umfang^2 / Fläche) für Rechtecke unterschiedlicher Seitenverhältnisse:

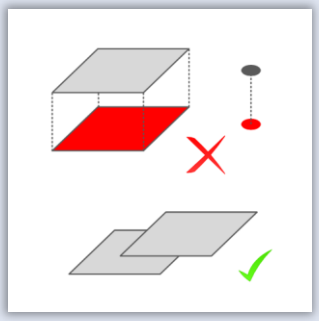
- Seitenverhältnis 1:1 (Quadrat): 16
- Seitenverhältnis 1:5: 28,8
- Seitenverhältnis 1:10: 48,4
- Seitenverhältnis 1:20: 88,2

Optional können Untersuchungsgebiete definiert werden, innerhalb dessen Lücken gefunden werden sollen. In diesem Fall werden einerseits auch Lücken zwischen den Polygonen und der Umgrenzung der Untersuchungsgebiete gemeldet, und andererseits werden Lücken außerhalb der Untersuchungsgebiete ignoriert. Damit kann beispielsweise die lückenlose Abdeckung von administrativen Einheiten überprüft werden.

Weitere Optionen dienen der Prüfung auf Lücken, welche kleiner sind als die XY-Toleranz des Raumbezugssystems, sowie zur Speicheroptimierung bei Prüfung sehr komplexer Polygonmengen.

Test: *QaNoGaps*

Keine Geometrie-Duplikate



Findet Features mit deckungsgleicher Geometrie. Als Duplikate gelten Features, deren Differenzgeometrie leer ist. Z- und M-Werte werden ignoriert, die XY-Toleranz des Raumbezugssystems wird angewendet.

Optional kann ein SQL-Ausdruck definiert werden, in welchem die Attribute zweier Features mit gleicher Geometrie verglichen werden können. In diesem Ausdruck können die deckungsgleichen Features mit „G1“ und „G2“ angesprochen werden. Ist dieser Ausdruck für die zwei Features erfüllt, dann gilt die Übereinstimmung der Geometrie als erlaubt und es wird kein Fehler gemeldet.

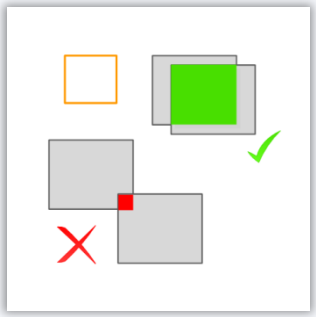
Beispiel: mit dem Ausdruck

```
G1.AdminLevel <> G2.AdminLevel
```

können flächengleiche administrative Einheiten erlaubt werden, sofern diese nicht zur selben administrativen Ebene gehören.

Test: *QaDuplicateGeometrySelf*

Minimaldimension für Überlappungsflächen

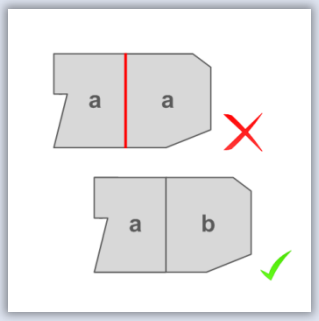


Findet Überlappungsflächen zwischen Polygonen, die kleiner sind als ein definierbarer Minimalwert.

Bei Polygon-Features, zwischen welchen Überlappungen grundsätzlich erlaubt sind, können mit diesem Test somit sehr kleine, inhaltlich nicht signifikante Überschneidungen erkannt werden.

Test: *QaMinIntersect, QaMinIntersectFactory*

Keine unnötigen Grenzen bei Polygonen mit gleichen Attributwerten



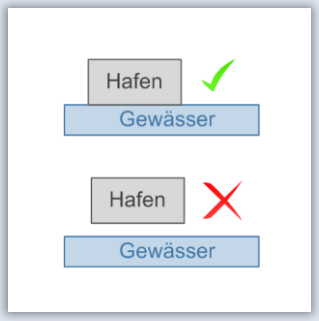
Findet Polygone, die sich berühren und für eine definierbare Liste relevanter Attribute identische Werte aufweisen.

Alternativ zur Liste kann ein SQL-Ausdruck definiert werden, in welchem die Attribute zweier sich berührender Polygone verglichen werden können. welche die involvierten Features erfüllen müssen. In diesem Ausdruck können die zwei Features mit „L“ bzw. „R“ angesprochen werden. Ist dieser Ausdruck für zwei Features erfüllt, dann wird kein Fehler gemeldet.

Optional können Polygone, die sich in nur einem Punkt berühren, ignoriert werden.

Test: *QaNeighbourAreas*

Erwartete Nachbarfeatures



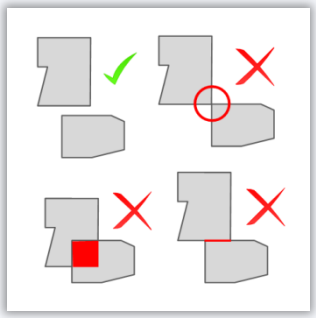
Findet Punkt-, Linien- oder Polygon-Features, die *kein* erwartetes Nachbarfeature berühren.

Optional kann ein SQL-Ausdruck definiert werden, in welchem die Attribute zweier sich berührenden Features verglichen werden können. In diesem Ausdruck können die berührenden Features mit „G1“ und „G2“ angesprochen werden. Ist dieser Ausdruck für die zwei Features nicht erfüllt, dann gilt die Berührung als nicht relevant. In diesem Fall wird ein Fehler gemeldet, sofern nicht eine weitere, relevante Berührung zu einem anderen Feature existiert.

Dieser Test existiert in zwei Varianten, einmal für den Vergleich zwischen Features aus zwei *unterschiedlichen* Mengen von Featureklassen, und einmal für den Vergleich zwischen allen Features innerhalb *einer* Menge von Featureklassen.

Test: *QaMustTouchSelf, QaMustTouchOther*

Keine Überschneidung oder Berührung (räumliche Beziehung „intersects“)



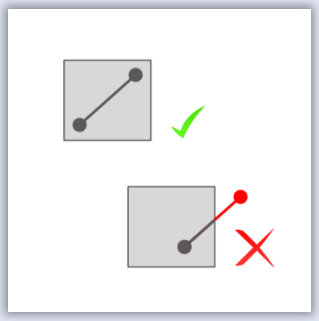
Findet Features, die andere Features in irgendeiner Form berühren oder überschneiden.

Ausnahmen können anhand eines SQL-Ausdrucks definiert werden, in welchem die Attribute zweier sich berührenden Features verglichen werden können. In diesem Ausdruck können die berührenden Features mit „G1“ und „G2“ angesprochen werden. Ist dieser Ausdruck für die zwei Features erfüllt, dann gilt der Kontakt der zwei Features als erlaubt und es wird kein Fehler gemeldet.

Dieser Test existiert in zwei Varianten, einmal für den Vergleich zwischen Features aus zwei *unterschiedlichen* Mengen von Featureklassen, und einmal für den Vergleich zwischen allen Features innerhalb *einer* Menge von Featureklassen.

Test: *QaIntersectsOther, QaIntersectsSelf*

Vollständige Umschließung (räumliche Beziehung „contains“)

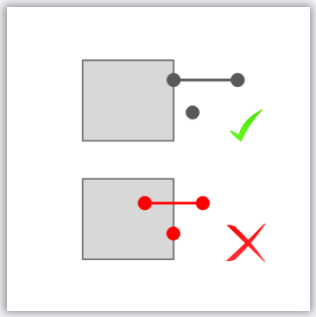


Findet Features, die nicht vollständig umschlossen werden von jeweils einem anderen Feature, und meldet die außerhalb liegenden Geometrieteile als Fehler.

Optional kann ein SQL-Ausdruck definiert werden, in welchem die Attribute des enthaltenen und des umschließenden Features verglichen werden können. In diesem Ausdruck kann das umschließende Feature mit „G1“ und das enthaltende Feature mit „G2“ angesprochen werden. Ist dieser Ausdruck für die zwei Features nicht erfüllt, dann gilt die Umschließung als nicht relevant. In diesem Fall wird ein Fehler gemeldet, sofern nicht ein weiteres, relevantes umschließendes Feature existiert.

Test: *QaContainsOther*

Keine Kreuzungen (räumliche Beziehung „crosses“)



Findet Features, die sich mit einem anderen Feature kreuzen. Anwendbar für Linie/Linie, Line/Polygon, Multipoint/Polygon und Multipoint/Linie.

Kriterien für „crosses“ sind:

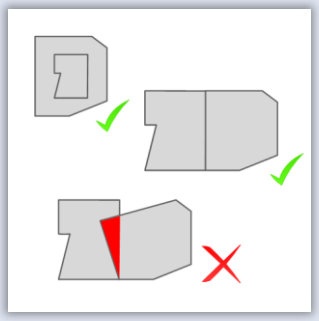
- Die Features schneiden jeweils den Außenbereich des anderen Features
- Die Dimension der Schnittgeometrie der Feature-Innenbereiche ist *geringer* als die maximale Dimension der zwei Features

Ausnahmen können anhand eines SQL-Ausdrucks definiert werden, in welchem die Attribute zweier sich kreuzenden Features verglichen werden können. In diesem Ausdruck können die kreuzenden Features mit „G1“ und „G2“ angesprochen werden. Ist dieser Ausdruck für die zwei Features erfüllt, dann gilt die Kreuzung als erlaubt und wird nicht als Fehler gemeldet.

Dieser Test existiert in zwei Varianten, einmal für den Vergleich zwischen Features aus zwei *unterschiedlichen* Mengen von Featureklassen, und einmal für den Vergleich zwischen allen Features innerhalb *einer* Menge von Featureklassen.

Test: *QaCrossesOther, QaCrossesSelf*

Keine Überlappung (räumliche Beziehung „overlaps“)



Findet Features, die sich mit einem anderen Feature „überlappen“. Anwendbar für Linie/Linie, Polygon/Polygon, Multipoint/Multipoint.

Kriterien für „overlaps“ sind:

- Die Features schneiden jeweils den Außenbereich des anderen Features
- Die Dimension der Schnittgeometrie der Feature-Innenbereiche ist *gleich* wie die der zwei Features

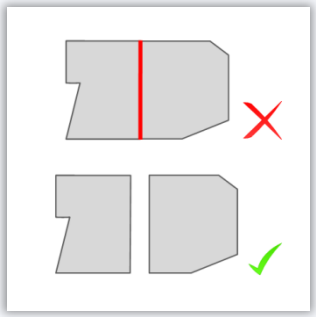
Somit entspricht die technische Definition von „overlaps“ nicht der umgangssprachlichen Definition für „Überlappung“. Beispielsweise ist für ein Polygon, das vollständig in einem anderen Polygon enthalten ist, die „overlaps“-Beziehung nicht erfüllt, da das enthaltene Polygon den Außenbereich des anderen Polygons nicht schneidet.

Ausnahmen können anhand eines SQL-Ausdruck definiert werden, in welchem die Attribute zweier sich „überlappenden“ Features verglichen werden können. In diesem Ausdruck können die „überlappenden“ Features mit „G1“ und „G2“ angesprochen werden. Ist dieser Ausdruck für die zwei Features erfüllt, dann gilt die Überlappung als erlaubt und wird nicht als Fehler gemeldet.

Dieser Test existiert in zwei Varianten, einmal für den Vergleich zwischen Features aus zwei *unterschiedlichen* Mengen von Featureklassen, und einmal für den Vergleich zwischen allen Features innerhalb *einer* Menge von Featureklassen.

Test: *QaOverlapsOther, QaOverlapsSelf*

Keine Berührung (räumliche Beziehung "touches“)



Findet Punkt, Multipoint-, Linien- oder Polygon-Features, welche andere Features berühren. Anwendbar für Polygon/Polygon, Linie/Linie, Linie/Polygon, Punkt/Linie, Multipoint/Linie.

Kriterien für „touches“ sind:

- Die Features sind nicht voneinander getrennt (ihre Schnittgeometrie ist nicht leer)
- Die Schnittgeometrie der *Innenbereiche* der zwei Features ist leer

Ausnahmen können anhand eines SQL-Ausdrucks definiert werden, in welchem die Attribute zweier sich berührenden Features verglichen werden können. In diesem Ausdruck können die berührenden Features mit „G1“ und „G2“ angesprochen werden. Ist dieser Ausdruck für die zwei Features erfüllt, dann gilt die Berührung als erlaubt und wird nicht als Fehler gemeldet.

Dieser Test existiert in zwei Varianten, einmal für den Vergleich zwischen Features aus zwei *unterschiedlichen* Mengen von Featureklassen, und einmal für den Vergleich zwischen allen Features innerhalb *einer* Menge von Featureklassen.

Test: *QaTouchesOther, QaTouchesSelf*

Keine räumliche Beziehung gemäß 9IM-Intersection-Matrix

Findet Features, die eine definierbare räumliche Beziehung mit einem anderen Feature aufweisen. Die Art der räumlichen Beziehung wird durch die 9-IM-Intersection-Matrix spezifiziert.

Optional können Ausnahmen für die aus der Matrix ableitbaren, potentiell mehrteiligen Schnittgeometrie zwischen zwei Features definiert werden. Dazu kann eine Liste von erlaubten Geometrietypen in dieser Schnittgeometrie definiert werden (punkt-, linien- oder flächenhafte Bestandteile der Schnittgeometrie). Es werden nur Fehler für Bestandteile der Schnittgeometrie gemeldet, deren Geometrietyp nicht in dieser Liste vorkommt.

Ausnahmen können anhand eines SQL-Ausdrucks definiert werden, in welchem die Attribute zweier gemäß Matrix in Beziehung stehender Features verglichen werden können. In diesem Ausdruck können die zwei Features mit „G1“ und „G2“ angesprochen werden. Ist dieser Ausdruck für die zwei Features erfüllt, dann gilt die räumliche Beziehung als erlaubt und wird nicht als Fehler gemeldet.

Dieser Test existiert in zwei Varianten, einmal für den Vergleich zwischen Features aus zwei *unterschiedlichen* Mengen von Featureklassen, und einmal für den Vergleich zwischen allen Features innerhalb *einer* Menge von Featureklassen.

Test: *QaIntersectionMatrixOther, QaIntersectionMatrixSelf*

Beispiel: Überschneidung der Innenbereiche (“interior intersects”)

9IM-Matrix		G2		
		I	B	E
G1	I	True	*	*
	B	*	*	*
	E	*	*	*

I: Interior (Innenbereich)
B: Boundary (Grenze)
E: Exterior (Außenbereich)

Die Zellen der Matrix werden zeilenweise aneinandergereiht, um den Matrix-Textstring zu bilden. Im Beispiel:

```
T*******
```

Räumliche Beziehung gemäß 9IM-Intersection-Matrix muss existieren

Findet Features, die *kein* Nachbarsfeature haben, für welches eine definierte 9IM-Intersection-Matrix erfüllt ist.

Optional können zusätzliche Regeln definiert werden für die Schnittgeometrie mit einem Nachbarfeature, welche der definierten Matrix entspricht. Dazu kann eine Liste von Geometrietypen in dieser Schnittgeometrie definiert werden (punkt-, linien- oder flächenhafte Bestandteile der Schnittgeometrie), die entweder *zwingend erforderlich* oder *nicht erlaubt* sind. Nur falls auch diese zusätzlichen Regeln erfüllt sind, gilt die räumliche Beziehung zum Nachbarfeature als erfüllt. Beispielsweise kann bei einer erforderlichen Berührung der Grenzen der Features zusätzlich gefordert werden, dass diese Berührung linienförmig sein muss, damit die Beziehung zum Nachbarfeature als relevant angeschaut wird.

Weitere Ausnahmen können anhand eines SQL-Ausdrucks definiert werden, in welchem die Attribute zweier gemäß der Intersection-Matrix in Beziehung stehender Features verglichen werden können. In diesem Ausdruck können die zwei Features mit „G1“ und „G2“ angesprochen werden. Das Feature-Paar gilt nur dann als in Beziehung stehend, wenn die Attributbedingung entweder undefiniert oder erfüllt ist.

Test: *QaMustIntersectMatrixOther*

Beispiel: Überschneidung von Geometrie-Grenzen

9IM-Matrix		G2		
		I	B	E
G1	I	*	*	*
	B	*	True	*
	E	*	*	*

I: Interior (Innenbereich)
B: Boundary (Grenze)
E: Exterior (Aussenbereich)

Die Zellen der Matrix werden zeilenweise aneinandergereiht, um den Matrix-Textstring zu bilden. Im Beispiel:

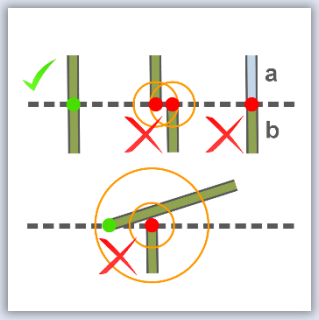
```
***T****
```

Grenzprüfung

Topologische und attributive Konsistenz entlang von Grenzen

Konsistente Fortsetzung grenzüberschreitender Linien

Neu



Findet Linienfeatures, welche auf der Grenze enden, die aber keine erwartete Fortsetzung auf der anderen Seite der Grenze haben.

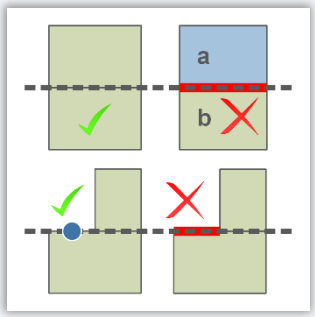
Regeln für die topologische und attributive Konsistenz eines zugeordneten Linien-Paars in den angrenzenden Gebieten können definiert werden, und ob eine Fortsetzung im Nachbargebiet erwartet wird. Attributregeln können als vergleichende SQL-Bedingung angegeben werden, und/oder als Liste von Feldern mit übereinstimmenden Werten (mit Optionen für das Ignorieren spezifischer Werte, und Unterstützung für mehrwertige Felder unter Verwendung eines Trennzeichens).

Grenzen können als Linien- oder Polygonfeatures vorliegen. Falls separate Grenz-Repräsentationen existieren pro Gebiet, wird in der Beschreibung zu einer gemeldeten Inkonsistenz angegeben, wenn die Grenzverläufe an der fraglichen Stelle nicht übereinstimmen

Test: *QaEdgeMatchCrossingLines*

Konsistente Fortsetzung grenzüberschreitender Flächen

Neu



Findet Polygonfeatures, welche die Grenze berühren, die aber keine erwartete Fortsetzung auf der anderen Seite der Grenze haben.

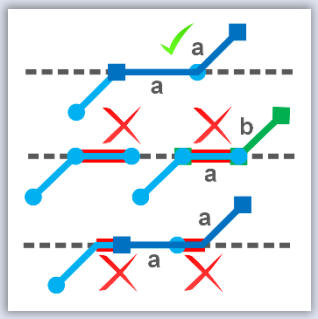
Regeln für die topologische und attributive Konsistenz eines zugeordneten Polygon-Paars in den angrenzenden Gebieten können definiert werden, und ob eine Fortsetzung im Nachbargebiet erwartet wird. Attributregeln können als vergleichende SQL-Bedingung angegeben werden, und/oder als Liste von Feldern mit übereinstimmenden Werten (mit Optionen für das Ignorieren spezifischer Werte, und Unterstützung für mehrwertige Felder unter Verwendung eines Trennzeichens).

Grenzen können als Linien- oder Polygonfeatures vorliegen. Falls separate Grenz-Repräsentationen existieren pro Gebiet, wird in der Beschreibung zu einer gemeldeten Inkonsistenz angegeben, wenn die Grenzverläufe an der fraglichen Stelle nicht übereinstimmen

Test: *QaEdgeMatchCrossingAreas*

Übereinstimmung grenzbildender Linien

Neu



Findet Linienfeatures, welche entlang der Grenze verlaufen, die aber keine erwartete Entsprechung im benachbarten Gebiet haben.

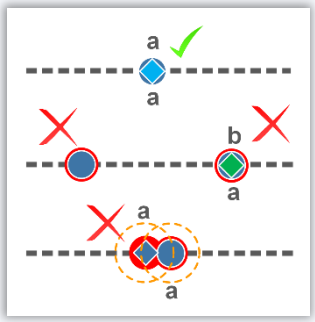
Regeln für die topologische und attributive Konsistenz eines zugeordneten Linien-Paars in den angrenzenden Gebieten können definiert werden, und ob ein entsprechendes Feature für das Nachbargebiet erwartet wird. Attributregeln können als vergleichende SQL-Bedingung angegeben werden, und/oder als Liste von Feldern mit übereinstimmenden Werten (mit Optionen für das Ignorieren spezifischer Werte, und Unterstützung für mehrwertige Felder unter Verwendung eines Trennzeichens).

Grenzen können als Linien- oder Polygonfeatures vorliegen. Falls separate Grenz-Repräsentationen existieren pro Gebiet, wird in der Beschreibung zu einer gemeldeten Inkonsistenz angegeben, wenn die Grenzverläufe an der fraglichen Stelle nicht übereinstimmen

Test: *QaEdgeMatchBorderingLines*

Übereinstimmung grenzbildender Punkte

Neu



Findet Punktfeatures, welche auf der Grenze liegen, die aber keine erwartete Entsprechung im benachbarten Gebiet haben.

Regeln für die topologische und attributive Konsistenz eines zugeordneten Punkt-Paars in den angrenzenden Gebieten können definiert werden, und ob ein entsprechendes Feature für das Nachbargebiet erwartet wird. Attributregeln können als vergleichende SQL-Bedingung angegeben werden, und/oder als Liste von Feldern mit übereinstimmenden Werten (mit Optionen für das Ignorieren spezifischer Werte, und Unterstützung für mehrwertige Felder unter Verwendung eines Trennzeichens).

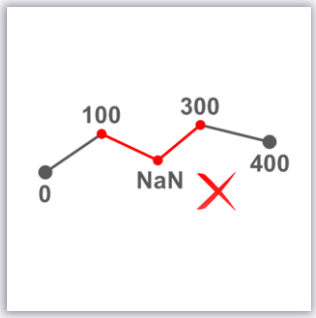
Grenzen können als Linien- oder Polygonfeatures vorliegen. Falls separate Grenz-Repräsentationen existieren pro Gebiet, wird in der Beschreibung zu einer gemeldeten Inkonsistenz angegeben, wenn die Grenzverläufe an der fraglichen Stelle nicht übereinstimmen

Test: *QaEdgeMatchBorderingPoints*

M-Werte

Bedingungen für die M-Werte von Features

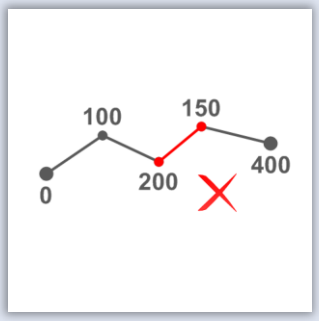
Gültige M-Werte



Findet M-Werte auf Linien-, Polygon-, Punkt, Multipoint- und Multipatch-Features, die nicht definiert sind (NaN), oder einen definierbaren ungültigen Wert aufweisen.

Test: *QaMeasures*

Monoton steigende oder fallende M-Werte



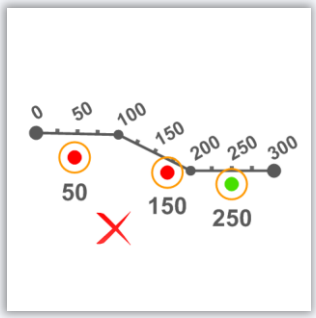
Findet Segmentfolgen auf Linien-Features, welche nicht monoton steigende oder fallende M-Werte aufweisen, d.h. deren M-Werte sich entgegen der erwarteten Richtung verändern. Dabei kann die erwartete Richtung der Monotonizität aufgrund der Digitalisierrichtung oder aufgrund der M-Werte der Endpunkte definiert werden.

Optional kann ein SQL-Ausdruck angegeben werden, mit welchem Features identifiziert werden, für welche die Richtung gegenüber der Digitalisierrichtung invertiert werden soll.

Außerdem können Segmentfolgen mit konstanten M-Werten optional erlaubt werden.

Test: *QaMonotonicMeasures*

Übereinstimmung von Linien-M-Werten mit Werten von Kalibrierungspunkten



Findet M-Werte entlang von Linien-Features, die sich vom Attribut- oder M-Wert eines benachbarten Punkt-Features unterscheiden. Es kann eine Toleranz für die maximal erlaubte Wertabweichung definiert werden.

Mit diesem Test können somit Stellen entlang von Linien identifiziert werden, bei denen die M-Werte nicht mehr der Kalibrierungsgrundlage entsprechen und ggf. neu berechnet werden sollten.

Für die Zuordnung von Punkten zu Linien-Feature kann eine Such-Distanz und optional ein SQL-Ausdruck definiert werden. In diesem Ausdruck kann der Punkt mit „P“ und die Linie mit „L“ angesprochen werden. Nur wenn der Ausdruck für einen Punkt und eine Linie erfüllt ist, dann ist der Soll-M-Wert des Punkt-Features für die Linie relevant.

Beispiel: Mit dem Ausdruck

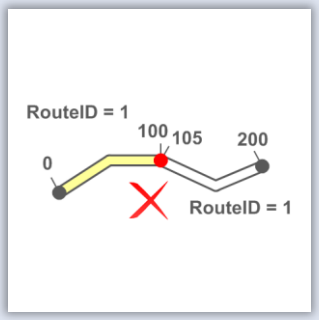
```
P.GEWAESSER_ID = L.GEWAESSER_ID
```

kann sichergestellt werden, dass nur Punkte desselben Gewässers für die Prüfung der Linien-M-Werte beigezogen werden.

Außerdem können Regeln definiert werden, wie der M-Wert auf der Linie bestimmt werden soll (nächster Vertex, nächstgelegene Stelle auf Linienverlauf), wie der Soll-M-Wert des Punkts bestimmt werden soll (Ableitung aus Attribut(en) oder direkt aus M-Wert).

Test: *QaMeasuresAtPoints*

Kontinuierliche M-Werte bei Linienverbindungen innerhalb einer Route



Findet Diskontinuitäten in den M-Werten an Verbindungsstellen von Linien-Features, die zur gleichen Route gehören.

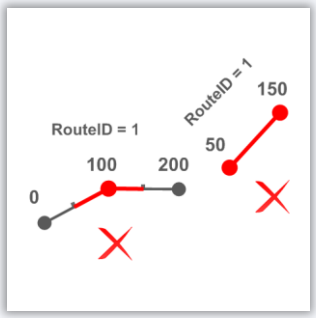
Verbundene Endpunkte von Linien, welche zur gleichen Route gehören (d.h. denselben Wert im Routen-ID-Attribut haben), müssen M-Werte aufweisen, welche sich um nicht mehr als die M-Toleranz des Raumbezugssystems unterscheiden.

Test: *QaRouteMeasuresContinuous*

M-Werte

Bedingungen für die M-Werte von Features

Eindeutige M-Werte innerhalb von Routen



Findet Segmentfolgen in Linien-Features derselben Route, welche identische M-Werte aufweisen. Solche nicht eindeutige M-Bereiche verhindern eine eindeutige Lokalisierung eines M-Werts innerhalb einer Route.

An Verbindungsstellen von Linien innerhalb einer Route sind Überlappungen der M-Werte innerhalb der M-Toleranz erlaubt. In allen anderen Fällen sind gleiche/überlappende M-Werte nicht erlaubt.

Als Fehler gemeldet werden die einzelnen, nicht eindeutigen M-Wertebereiche. Deren Fehlergeometrie kann sich über mehrere Features erstrecken.

Test: *QaRouteMeasuresUnique*

Z-Werte

Bedingungen für die Z-Werte von Features

Maximaler Z-Unterschied innerhalb einer Geometrie

Findet Linien-, Polygon-, Multipoint- oder Multipatch-Features, deren maximale und minimale Z-Werte sich um mehr als eine definierbare Toleranz voneinander unterscheiden.

Damit kann zum Beispiel geprüft werden, ob stehende Gewässer konstante Z-Werte aufweisen.

Im Fall eines Fehlers werden jene Stützpunkte gemeldet, welche außerhalb der halben Toleranz vom häufigsten Z-Wert liegen.

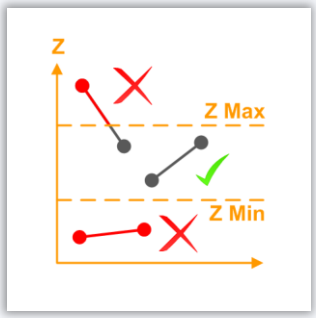
Test: *Qa3dConstantZ*

Höhenwerte innerhalb eines Z-Bereichs

Findet Punkt-, Linien-, Polygon-, Multipoint- und Multipatch-Features mit Z-Werten außerhalb eines erlaubten Z-Bereichs. Ist ein Z-Wert kleiner als die definierbare Z-Untergrenze oder grösser als die definierbare Obergrenze, gilt er als Fehler.

Bei Linien und Polygonen werden zusammenhängende Folgen von Segmenten/Segmentteilen, welche außerhalb derselben Bereichsgrenze liegen, als jeweils ein Fehler gemeldet. Bei Multipoint- und Multipatch-Features wird ein Fehler für alle Punkte außerhalb derselben Bereichsgrenze gemeldet.

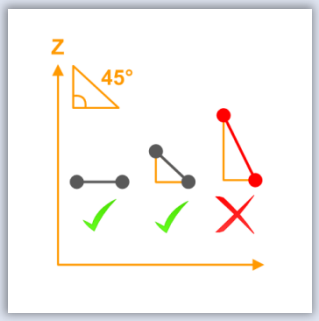
Test: *QaWithinZRange*



Maximaler Neigungswinkel

Findet Segmente von Linien- oder Polygon-Features, deren Neigung steiler ist als ein definierbarer Maximalwinkel.

Tests: *QaMaxSlope, QaMaxSlopeFactory*



Monoton steigende oder fallende Z-Werte

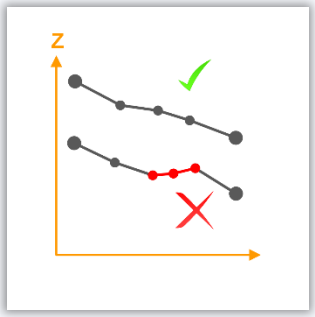
Neu

Findet Segmentfolgen auf Linien-Features, welche nicht monoton steigende oder fallende Z-Werte aufweisen, d.h. deren Z-Werte sich entgegen der erwarteten Richtung verändern. Dabei kann die erwartete Richtung der Monotonizität aufgrund der Digitalisierrichtung oder aufgrund der Z-Werte der Endpunkte definiert werden.

Optional kann ein SQL-Ausdruck angegeben werden, mit welchem Features identifiziert werden, für welche die Richtung gegenüber der Digitalisierrichtung invertiert werden soll.

Außerdem können Segmentfolgen mit konstanten Z-Werten optional erlaubt werden.

Test: *QaMonotonicZ*

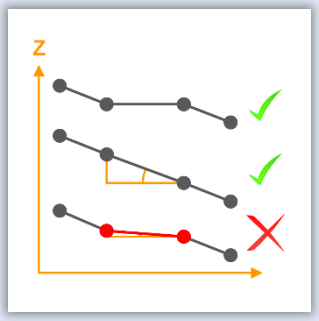


Horizontale Segmente

Neu

Findet Segmente in Polygon-, Linien- und MultiPatch-Features, die annähernd, aber nicht ausreichend horizontal verlaufen.

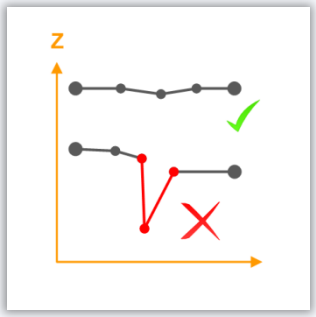
Test: *QaHorizontalSegments*



Maximale Kurvatur

Findet Segmente von Linien- oder Polygon-Features, bei denen der Profilverlauf im Kontext der Nachbarssegmente eine zu hohe Kurvatur aufweist (d.h. eine zu rasche Veränderung des Neigungswinkels).

Test: *QaSmooth*

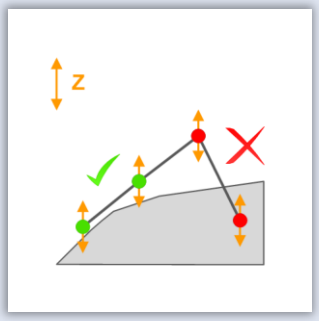


Abweichung von Geländehöhe (Stützpunkte)

Findet Punkte oder Stützpunkte, deren Z-Werte um mehr als einen definierbaren Wert von der Höhe eines Geländemodells (Raster-, Mosaic- oder Terrain-Dataset) abweichen.

Alternativ kann auch umgekehrt geprüft werden, dass die Punkte einen Minimalabstand vom Gelände einhalten. Dabei kann festgelegt werden, ob sie nur oberhalb, nur unterhalb, oder sowohl oberhalb als auch unterhalb des Geländes liegen dürfen.

Test: *Qa3dSmoothing*



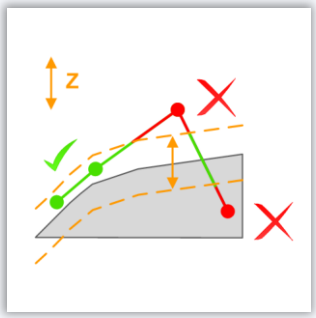
Abweichung von Geländehöhe (Linienverläufe)

Findet Folgen von Segmenten bzw. Segmentteilen von Linien- und Polygon-Features, welche außerhalb eines Toleranzbereichs um die Oberfläche eines Geländemodells (Raster-, Mosaic- oder Terrain-Dataset) liegen.

Alternativ kann auch umgekehrt geprüft werden, dass die Segmentverläufe einen Minimalabstand von der Geländeoberfläche einhalten. Dabei kann festgelegt werden, ob sie nur oberhalb, nur unterhalb, oder sowohl oberhalb als auch unterhalb des Geländes liegen dürfen.

Optional kann bei Linien ein definierbarer Bereich um die Endpunkte ignoriert werden. Damit kann zum Beispiel geprüft werden, dass Brücken einen Minimalabstand vom Gelände einhalten, ohne dass der Bereich um die Brückenköpfe jeweils als Fehler gemeldet würde.

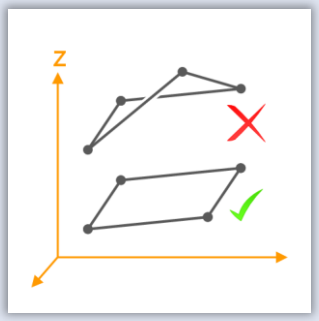
Test: *Qa3dPipe*



Koplanare Ringe

Findet Multipatch- oder Polygonringe, deren Stützpunkte nicht koplanar sind, das heißt nicht in einer Ebene liegen.

Test: *QaCoplanarRings*



Minimaler Z-Abstand zwischen Features

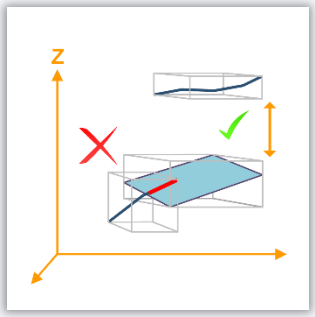
Neu

Findet Features, welche sich in 2D schneiden, und zwischen denen ein ungenügender Z-Abstand existiert. Der Z-Abstand wird aufgrund der umschliessenden 3D-Boxen der Features bestimmt.

Zusätzlich kann eine SQL-Bedingung definiert werden, mit welchem die Attributwerte der beteiligten Features verglichen werden können. Das gemäß minimalem Z-Wert höher gelegene Feature kann dabei mit dem Alias „U“ angesprochen werden, das tiefer gelegene Feature mit „L“. Mit dem Ausdruck `U.STUFE > L.STUFE` kann beispielsweise geprüft werden, dass das höherliegende Feature einer höheren „Stufe“ zugeordnet ist als das darunterliegende Feature.

Dieser Test existiert in zwei Varianten, einmal für den Vergleich zwischen Features aus zwei *unterschiedlichen* Mengen von Featureklassen, und einmal für den Vergleich zwischen allen Features innerhalb *einer* Menge von Featureklassen.

Test: *QaZDifferenceOther, QaZDifferenceSelf*



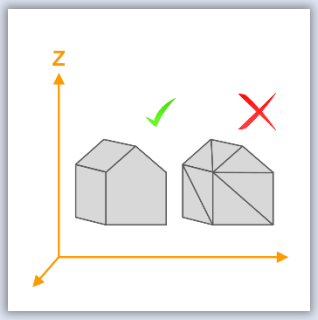
Einschränkung der erlaubten Teil-Geometrietypen

Neu

Findet MultiPatch-Features, welche bestimmte unerwünschte Teil-Geometrietypen enthalten.

MultiPatch-Features können aus einer Kombination von Ringen, Dreiecksfächern, Dreiecksstreifen, und Dreiecks-Patches bestehen. In gewissen Situationen (z.B. bei Verwendung bestimmter Bearbeitungswerkzeuge) können einige dieser Geometrietypen unerwünscht sein. Die erlaubten Geometrietypen können individuell festgelegt werden.

Test: *QaMpAllowedPartTypes*



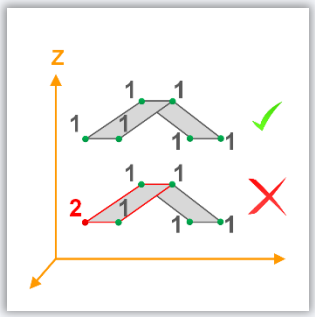
Konstante Punkt-IDs pro Ring

Neu

Findet MultiPatch-Features, bei welchen die Punkt-ID's innerhalb eines Rings nicht konstant sind.

Gewisse MultiPatch-Editierwerkzeuge verwenden Punkt-ID's, um Ringe in übergeordnete Strukturen zu gruppieren. Diese Werkzeuge erfordern, dass die Punkt-ID's innerhalb eines Rings konstant sind.

Test: *QaMpConstantPointIdsPerRing*



Keine Löcher in 2D-Grundrissen von MultiPatches

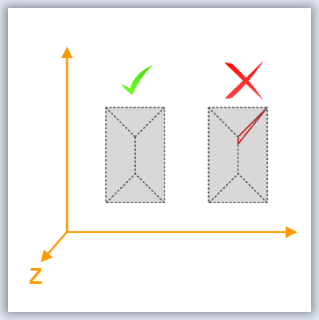
Neu

Findest MultiPatch-Features, deren 2D-Grundrisse Löcher enthalten.

Optional können alle, oder alle horizontalen, inneren Ringe erlaubt werden. Außerdem können Löcher, welche eine definierte Fläche überschreiten, erlaubt werden.

Um sehr kleine Löcher zu erkennen, kann der Test eine XY-Resolution/Toleranz verwenden, welche um einen definierbaren Faktor reduziert ist.

Test: *QaMpFootprintHoles*



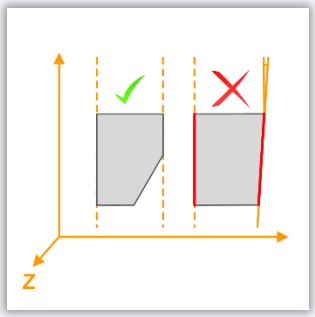
Konsistente Azimuth-Winkel horizontaler MultiPatch-Segmente

Neu

Findet Paare von horizontalen Segmenten in MultiPatch-Features, deren Azimuth-Winkel annähernd, aber nicht ausreichend gleich sind.

Der Segment-Vergleich kann sich auf das gesamte Feature beziehen, oder separat auf jeden MultiPatch-Ring.

Test: *QaMpHorizontalAzimuths*

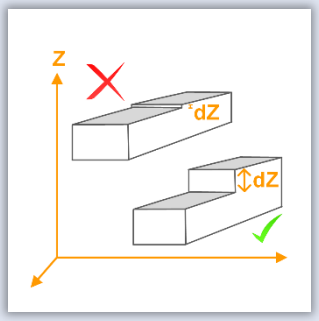


Konsistente Z-Werte horizontaler MultiPatch-Segmente

Neu

Findet Paare von horizontalen Segmenten in MultiPatch-Features, deren Z-Werte annähernd, aber nicht ausreichend gleich sind.

Test: *QaMpHorizontalHeights*

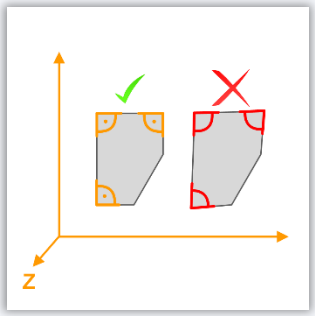


Konsistente rechte Winkel zwischen horizontalen MultiPatch-Segmenten

Neu

Findet Paare von horizontalen Segmenten in MultiPatch-Features, die fast, aber nicht ausreichend senkrecht zueinanderstehen. Optional kann der Vergleich eingeschränkt werden auf verknüpfte bzw. in der Nähe liegende Segmente.

Test: *QaMpHorizontalPerpendicular*



Keine Überschneidung zwischen 2D-Grundrissen von Ringen innerhalb eines MultiPatch-Feature

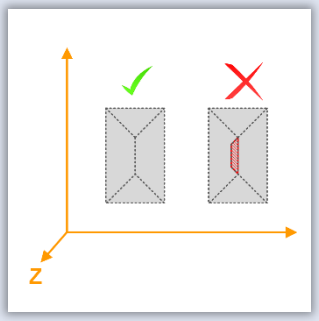
Neu

Findet Überschneidungen zwischen 2D-Grundrissen von Ringen innerhalb desselben MultiPatch-Features.

Optional können Überschneidungen erlaubt werden zwischen Ringen, deren Stützpunkte alle denselben Punkt-ID-Wert aufweisen, und wo dieser konstante Wert für die überschneidenden Ringe unterschiedlich ist.

Um sehr kleine Überschneidungen zu erkennen, kann der Test eine XY-Resolution/Toleranz verwenden, welche um einen definierbaren Faktor reduziert ist.

Test: *QaMpNonIntersectingRingFootprints*



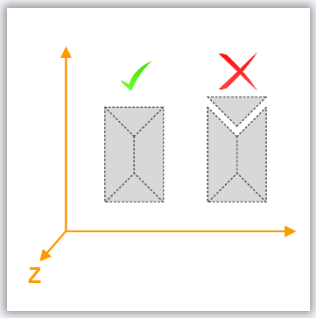
Keine mehrteilige 2D-Grundrisse von MultiPatch-Features

Neu

Findet MultiPatch-Features, deren 2D-Grundrisse aus mehreren räumlich getrennten Teilen bestehen.

Um sehr kleine getrennte Geometrieteile zu erkennen, kann der Test eine XY-Resolution/Toleranz verwenden, welche um einen definierbaren Faktor reduziert ist.

Test: *QaMpSinglePartFootprint*

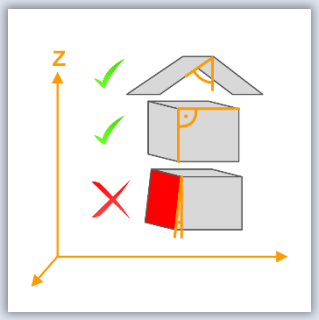


Konsistente senkrechte Wände in MultiPatch-Features

Neu

Findet Flächen in MultiPath-Features, welche annähernd, aber nicht ausreichend senkrecht sind.

Test: *QaMpVerticalFaces*



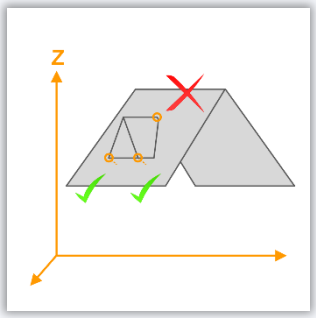
Keine kleinen Z-Abweichungen zwischen Stützpunkten und MultiPatch-Flächen

Neu

Findet Stützpunkte, deren Z-Werte in nahe bei, aber nicht ausreichend genau auf einem MultiPatch-Ring oder Dreieck liegen. Die Stützpunkte beliebiger Featureklassen können so geprüft werden.

Separate Annäherungstoleranzen können angegeben werden für Stützpunkte oberhalb bzw. unterhalb einer MultiPatch-Fläche. Die Behandlung nicht ausreichend koplanarer Ringe kann definiert werden. Optional können Flächen, welche um weniger als ein definierter Winkel geneigt sind, ignoriert werden.

Test: *QaMpVertexNotNearFace*

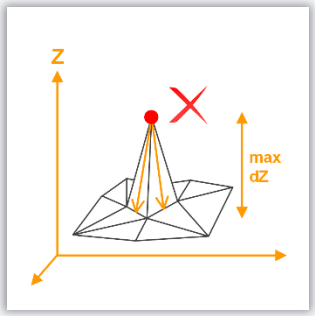


Keine „Spikes“

Neu

Findet Knoten in Terrain-Datasets, bei welchen die Z-Abstände zu den benachbarten Knoten sowie die Hangneigung der angrenzenden Dreiecke alle einen definierten Maximalwert überschreiten.

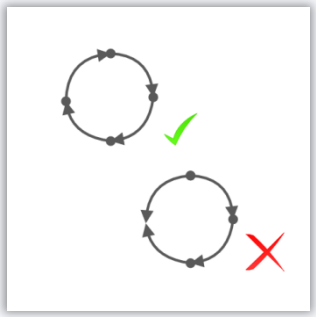
Test: *QaTerrainSpikes*



Polygonnetzwerke

Bedingungen für Flächen aus Grenzlinien und Zentroiden

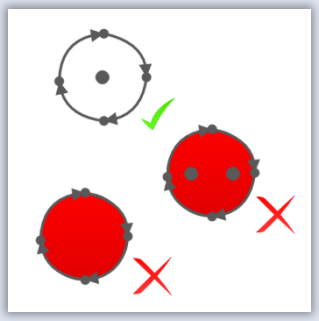
Ungültige Ringe



Überprüft, ob Linien- Features geschlossene, konsistent orientierte Ringe bilden. Linien-Features, die nicht zu einem gültigen Ring gehören, werden als Fehler gemeldet. Dabei kann angegeben werden, ob die Linienfolgen, welche einen Ring bilden, im Uhrzeigersinn oder im Gegenuhrzeigersinn orientiert sein müssen.

Test: *QaBorderSense*

Zentroide



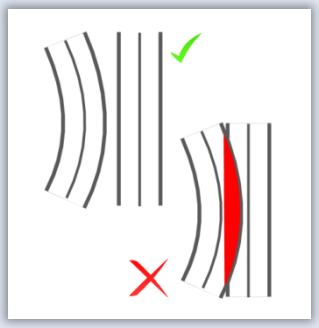
Überprüft, ob von Linien-Features umschlossene Flächen je genau ein Punkt-Feature enthalten. Zusätzlich können die Attribute auf einer Grenzlinie mit jenen der benachbarten Zentroide verglichen werden. Dazu kann ein SQL-Ausdruck definiert werden, in welchem die Grenzlinie als „B“, der links der Linie gelegene Zentroid als „L“ und der rechts gelegene Zentroid als „R“ angesprochen werden können. Damit kann beispielsweise geprüft werden, dass die Nachbar-Zentroide der Grenzlinie eines bestimmten Typs gewisse Attribute gleich oder ungleich haben.

Test: *QaCentroids*

Symbolisierung

Bedingungen für kartographische Symbolisierung

Keine graphischen Konflikte



Findet Überlappungen in der Symbolisierung von Features (grafischer Konflikt).

Trotz korrekter geometrischer Situation von Features, kann es bei der Symbolisierung zu grafischen Konflikten kommen, die aus kartografischer Sicht fehlerhaft sind.

Der Test prüft Repräsentationen für einen definierbaren Referenzmaßstab auf Symbolüberlappungen. Dazu verwendet er das Geoverarbeitungs-Werkzeug „Grafikkonflikt ermitteln“. Die ermittelte Überlappungsfläche stellt die Fehlergeometrie dar.

Es kann eine Konfliktentfernung angegeben werden, die sicherstellt, dass Minimalabstände zwischen den Symbolen eingehalten werden.

Test: *QaGraphicConflict*

Keine ungültigen Repräsentationen

Neu

Findet Features deren Repräsentation nicht den definierten Kriterien entspricht. Dabei kann geprüft werden, ob die Repräsentation sichtbar ist, eine freie Repräsentation ist, eine gültige Repräsentationsregel verwendet oder Repräsentations-Overrides verwendet.

	ID	RuleID	Override
X	1	-1	<null>
	2	1	<null>
X	3	1	BLOB

Optional können Ausnahmen für die Prüfung von Repräsentations-Overrides angegeben werden um gezielt bestimmte Overrides zu erlauben. Es kann beispielsweise erlaubt sein die Ausrichtung eines Repräsentationsmarkes zu ändern, während andere Overrides nicht erlaubt sind.

Test: *QaRepresentationConstraints*

Schema

Bedingungen für Schemas von Tabellen/Featureklassen

Keine ungültigen Feld-Aliasnamen

Prüft die Aliasnamen der Felder einer Tabelle. Dabei können Regeln für die maximale Länge des Aliasnamens, für die Groß-/Kleinschreibung und für erwartete Unterschiede vom Feldnamen definiert werden. Bei Systemfeldern kann definiert werden, ob benutzerdefinierte Aliasnamen erlaubt sind, die dann wiederum den definierten Regeln unterworfen sind. Optional kann geprüft werden, ob die Aliasnamen innerhalb der Tabelle eindeutig sind.

Test: *QaSchemaFieldAliases*

Keine ungültigen Feldnamen

Prüft die Feldnamen einer Tabelle. Dabei können Regeln für die maximale Länge des Feldnamens und für die Groß-/Kleinschreibung definiert werden. Außerdem kann geprüft werden, dass die Feldnamen innerhalb der ersten, definierbaren Anzahl Zeichen eindeutig sind. Dies ist beispielsweise beim Export nach Shapefile relevant, da dort die Feldnamen nach zehn Zeichen abgeschnitten werden.

Test: *QaSchemaFieldNames*

Reguläre Ausdrücke für Feldnamen

Prüft, ob die Feldnamen einer Tabelle einem definierbaren regulären Ausdruck entsprechen.

Damit kann beispielsweise geprüft werden, dass Feldnamen keine Umlaute oder Spezialzeichen enthalten.

Test: *QaSchemaFieldNameRegex*

Keine Felder mit reservierten Feldnamen

Prüft, ob die Namen der Felder einer Tabelle einem reservierten, nicht erlaubten Namen entsprechen. Diese reservierten Namen können für den Test aufgelistet oder aus einer externen Tabelle gelesen werden. Bei der Speicherung der reservierten Namen in einer Tabelle können außerdem der Grund für die Reservierung des Namens und ggf. ein alternativer, gültiger Name, welcher stattdessen verwendet werden sollte, angegeben werden.

Test: *QaSchemaReservedFieldNames*

Erwartete Feldeigenschaften

Prüft, ob ein definierbares Feld einer Tabelle die erwarteten Eigenschaften besitzt. Dabei werden Datentyp, Länge, Alias- oder Domänenname untersucht. Außerdem kann angegeben werden, ob das Feld in der Tabelle vorhanden sein *muss*.

Test: *QaSchemaFieldProperties*

Felder referenzieren Domänen mit einem kompatiblen Datentyp

Prüft, ob die von den Feldern einer Tabelle referenzierten Domänen zum Datentyp des Felds passen.

Test: *QaSchemaFieldDomains*

Keine ungültigen Domänennamen.

Prüft die Namen der von einer Tabelle verwendeten Domänen. Dabei können Regeln für die maximale Länge des Domänennamens und für die Groß-/Kleinschreibung definiert werden. Außerdem kann optional verlangt werden, dass der Domänenname den Namen des Feldes enthält, von dem die Domäne referenziert wird, und dass der Domänenname ein erwartetes Präfix enthält.

Test: *QaSchemaFieldDomainNames*

Keine ungültigen Domänenbeschreibungen

Prüft die Beschreibungen von allen in einer Tabelle referenzierten Domänen. Dabei kann eine maximale Länge des Beschreibungstexts angegeben werden. Optional kann geprüft werden, dass die Domänenbeschreibung innerhalb des Workspace der geprüften Tabelle, oder einem anderen Workspace eindeutig ist.

Test: *QaSchemaFieldDomainDescriptions*

Schema

Bedingungen für Schemas von Tabellen/Featureklassen

Reguläre Ausdrücke für Domänennamen

Prüft, ob die Namen der von einer Tabelle verwendeten Domänen einem definierbaren regulären Ausdruck entsprechen.

Damit kann beispielsweise geprüft werden, dass Domänennamen keine Umlaute oder Spezialzeichen enthalten.

Test: *QaSchemaFieldDomainNameRegex*

Codierte Wertdomänen mit gültigen Wertlisten

Prüft die Wertelisten aller codierten Wertdomänen, die von einer Tabelle verwendet werden. Dabei können Regeln für die maximale Länge der Wert-Namen, für deren Eindeutigkeit innerhalb der Domäne, und für die minimale Anzahl von Werten in der Domäne definiert werden. Außerdem kann definiert werden, dass eine gewisse Mindestzahl von Wert-Namen ungleich dem Wert-Code sein müssen.

Test: *QaSchemaFieldDomainCodedValues*

Identisches räumliches Bezugssystem

Prüft, ob das Raumbezugssystem einer Featureklasse identisch ist mit einem Referenz-Raumbezugssystem. Dieses Referenzsystem kann anhand einer Referenz-Featureklasse oder eines Xml-Strings angegeben werden. Dabei kann angegeben werden, ob zusätzlich zum horizontalen Koordinatensystem auch das vertikale Koordinatensystem gleich sein muss, und ob die Einstellungen für die Koordinatenspeicherung (Ursprungspunkt und Präzision von XY, Z, und M-Werten) und die entsprechenden Toleranzen gleich sein müssen.

Test: *QaSchemaSpatialReference*