GEOWERKSTATT

Erneuerung BauGK GIS Schnittstelle

Spezifikation der CCC-Schnittstelle 1.0

Schnittstellenbeschreibung, SO!GIS

9. Mai 2018

Auftragnehmer

GeoWerkstatt GmbH
Bleichemattstrasse 2
5000 Aarau

Auftraggeber

Amt für Geoinformation
Rötistrasse 4
4501 Solothurn

Dokumentinformationen

Projekttitel	Erneuerung BauGK GIS Schnittstelle
Dokumenttitel	Spezifikation der CCC-Schnittstelle 1.0
Dateiname	Spezifikation_CCC_Schnittstelle.docx
Seitenzahl	22
Letzte Bearbeitung am	9. Mai 2018, 16:11
Letzte Bearbeitung durch	Oliver Jeker

Änderungsnachweis

Version	Status	Datum	Bearbeiter	Änderung / Bemerkung
0.1	In Bearbeitung	03.11.2017	TG	Initialversion
0.9	In Bearbeitung	15.11.2017	TG	Anpassungen nach erstem Review
0.91	In Bearbeitung	20.11.2017	TG	Anpassungen nach zweitem Review
0.92	In Bearbeitung	01.12.2017	TG	Neue Nachricht *dataWritten*, *ready* wird an beide Applikationen geschickt, Timeout bei Verbindungsaufbau.
0.95	In Bearbeitung	07.03.2018	OJ	Kapiteltrennung Handshake- und Payloadoperationen, Erläuterung der Operationen mit Objektkatalog. Context immer als Objekt.
0.96	In Bearbeitung	20.03.2018	OJ	Ergänzung Handshake-Operationen mit clientName Streichung von apiVersion in Payload-Operationen Operation selected: Umbenennung properties zu context_list
0.97	In Bearbeitung	27.03.2018	OJ	Explizite error-Operation
1.0	Freigegeben	02.04.2018	OJ	Verwendung expliziterer Namen für die Operationen. Eingliederung der relevanten Fussnoten in den Text.

INHALTSVERZEICHNIS

CLIENT-CLIENT CONTEXT (CCC) SCHNITTSTELLE 5

1. EINLEITUNG 5
2. SYSTEMÜBERSICHT 5
3. CCC-NACHRICHT 6
   - 3.1 apiVersion 6
   - 3.2 method 6
   - 3.3 session 6
   - 3.4 clientName 6
   - 3.5 context 7
   - 3.6 zoomTo 7
   - 3.7 data 7
   - 3.8 notifyError 8
   - 3.9 Beispiele 10
4. OPERATIONEN 11
   - 4.1 Handshake-Operationen 11
   - 4.2 Payload-Operationen 12
   - 4.3 NotifyError-Operation 18
5. CLIENT-CLIENT CONTEXT SERVICE 20
6. SICHERHEIT 21

CLIENT-CLIENT CONTEXT (CCC) SCHNITTSTELLE

1. EINLEITUNG

Im *Konzept «Integration von Fach- und Geometriedaten»* wurden verschiedene Muster erarbeitet wie sich GIS-Aspekte in Fachapplikationen integrieren lassen. Dieses Dokument beschreibt die für das *Muster „AGI-WebGIS-Client" ausschliesslich für die Geometriedaten* benötigte Client-Client Context (CCC) Schnittstelle.

2. SYSTEMÜBERSICHT

Zentraler Gedanke des Protokolls ist, dass die Datenhoheit bei der Fachanwendung liegt. Dies bedeutet, dass das WebGIS Änderungen nie direkt speichert, sondern stets in Form von notifyEditGeoObjectDone-Nachrichten an die Fachanwendung schickt. Diese ist dann dafür verantwortlich, den GIS-Cache, falls notwendig, zu aktualisieren.

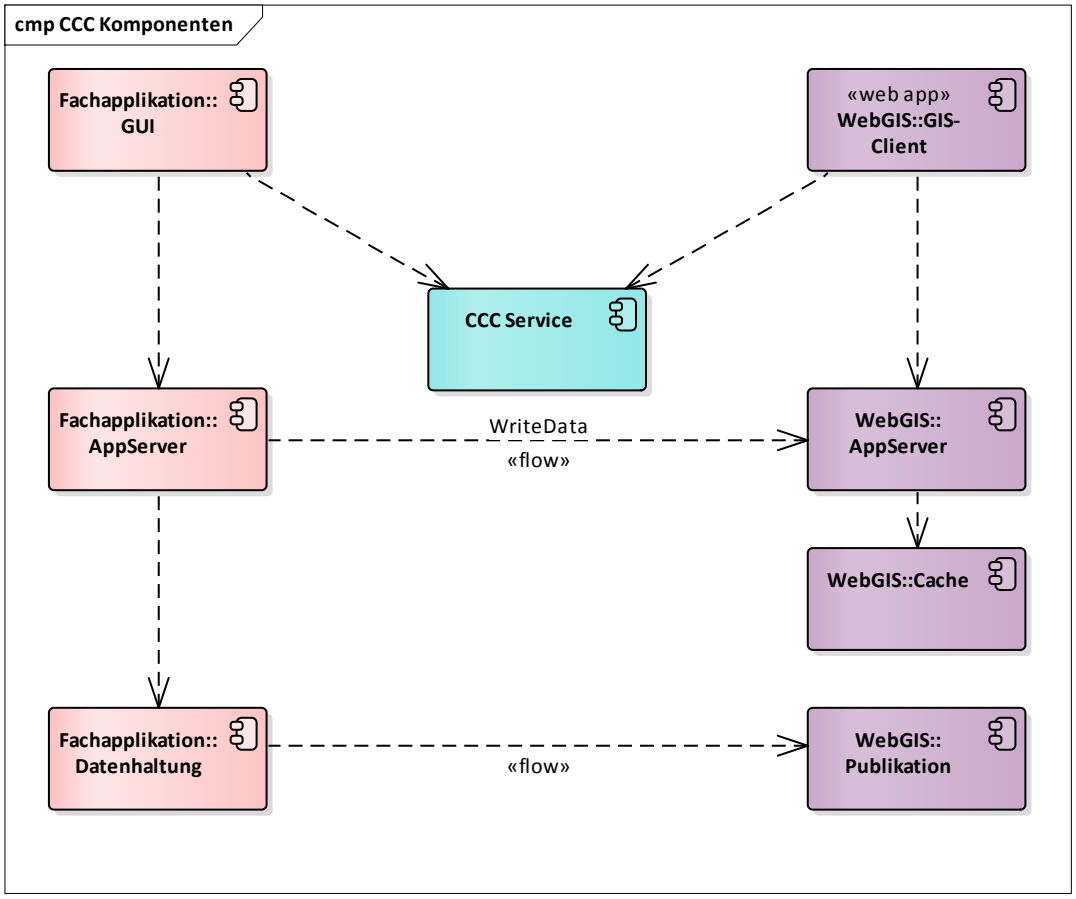

Abbildung 1 Die an der Kommunikation über die CCC-Schnittstelle beteiligten Komponenten

Das Protokoll beschreibt die Art wie eine Fachapplikation mit dem dazugehörenden WebGIS-Client kommuniziert. Darauf aufbauend muss für jede Anwendung festgelegt werden, was die Nachrichten im Kontext der konkreten Anwendung bedeuten. Typischerweise werden dabei die folgenden Punkte spezifiziert:

- Wie die WebGIS-Anwendung aufgerufen wird,

- wie *session* zwischen Fachanwendung und WebGIS übertragen wird,
- ob die WebGIS-Anwendung context auswerten muss, und falls ja, wie,
- Grundkarte, Layer und initialer Kartenausschnitt der WebGIS-Anwendung,
- wie *zoomTo* von der WebGIS-Anwendung interpretiert werden soll,
- was in der WebGIS-Anwendung erfasst oder ausgewählt werden kann,
- wie die GIS-Objekte als GeoJSON data übertragen werden, und
- ob und wie die Daten an den GIS-Cache übergeben werden.

3. CCC-NACHRICHT

Eine CCC-Nachricht ist eine WebSocket Nachricht, die zwischen dem CCC-Service und der Fachapplikation bzw. dem WebGIS Client ausgetauscht wird, und enthält genau ein JSON-Objekt. Namen und Typen der Felder sind den Beschreibungen in Googles JSON Styleguide angelehnt (https://google.github.io/styleguide/jsoncstyleguide.xm).

Abgesehen von Kontrollnachrichten (Session- und Fehlerbehandlung) beziehen sich die Nachrichten auf jeweils ein Fachobjekt (ein Objekt mit aus fachlicher Sicht gleichen Fachattributen), und können kein, ein oder mehrere GIS-Objekte beschreiben. Falls beispielsweise im GIS mehrere Polygone zu einem Fachobjekt gehören werden diese jeweils gemeinsam in einer CCC-Nachricht übertragen.

3.1 apiVersion

Beschreibt die Version der Schnittstelle, nach der die Nachricht beschrieben ist. In dieser Version ist der Wert stets "1.0", falls andere Werte gelesen werden muss die Nachricht verworfen und ggf. eine Fehlermeldung zurückgeschickt werden.

```
{ "apiVersion": "1.0" }
```

3.2 method

Der Name der Operation, die ausgeführt werden soll.

```
{ "method": "editGeoObject" }
```

3.3 session

Die Session ID. Die Session ID wird vom Fachapplikation-Client erstellt und beim Initialisieren der Verbindung an das WebGIS übergeben. Beide verwenden diese ID in der jeweiligen **Fehler! Verweisquelle konnte nicht gefunden werden.**-Operation. Der CCC-Server benutzt diese ID, um Fachanwendung und WebGIS einander zuzuordnen und die Nachrichten entsprechend weiterzuleiten.

```
{ "session": "{E9C62508-025A-4A0F-B342-5A632282ABD8}" }
```

3.4 clientName

Sprechender Name des Fach- oder GIS-Clients welcher als Endpunkt in einer CCC-Session enthalten ist. Wird zwecks Fehlereingrenzung im Betrieb verwendet, zum Beispiel zwecks Anzeige in logs.

```
{ "clientName": "Axioma Mandant AfU" }
```

3.5 context

Die Fachapplikation kann den Context benutzen um Anfragen und Antworten zu korrelieren, beschreibt typischerweise das im GIS behandelte Fachobjekt. Das WebGIS muss diesen Wert bei Nachrichten, die sich auf dasselbe Objekt beziehen, zurücksenden.

```
{
	"context": {
		"zonen_num": CH493287040689
	}
}
```

Innerhalb des context-Objekts können neben Zahlen und Strings auch Arrays oder andere JSON-Objekte verwendet werden. Der Regelfall dürften Name/Wert-Paare wie im folgenden Beispiel sein.

```
{
	"context": {
		"bfs_num": 231
		"parz_num": 1951
	}
}
```

3.6 zoomTo

Mit diesem Wert spezifiziert die Fachapplikation bei createGeoObject-Operationen welcher Kartenausschnitt gewählt werden soll. Hat dieses Feld den Wert *null*, wird in der Regel der über die Grundkarte konfigurierte Ausschnitt oder der letztgewählte Ausschnitt verwendet.

Die Interpretation dieser Angaben muss zwischen Fachapplikation und WebGIS-Anwendung abgesprochen werden. Da die Fachanwendung möglichst wenig GIS-Wissen haben soll, ist dies in der Regel weder ein Extent noch direkt ein GIS-Layer und eine Objekt-ID. Da es sich um eine räumliche Einschränkung handelt, die der Fachanwendung bekannt sein muss, bieten sich übergreifend definierte institutionelle Gliederungen wie beispielsweise Gemeinde oder Bezirk (mit BFS-Nummer) an.

```
{
	"zoomTo": [
		{
			"nbident": "SO0200002513",
			"nummer": 3765
		},
		{
			"nbident": "SO0200002422"
		}
	]
}
```

3.7 data

Dieses Feld enthält die eigentlichen Geometriedaten als ein Objekt im GeoJSON Format, wobei alle Koordinatenzahlen das CH1903+/LV95 Referenzsystem verwenden.

Die CCC-Schnittstelle als Transportprotokoll macht keine Einschränkungen bezüglich den eingesetzten GeoJSON-Geometrietypen. *Point*, *MultiPoint*, *LineString*, *MultiLineString*, *Polygon* und *GeometryCollection* sind alle vorstellbar, ebenso die Verwendung von GeoJSON Features Feature Collections. Welche davon tatsächlich eingesetzt werden muss pro Anwendung zwischen Fachanwendung und der entsprechenden WebGIS-Anwendung vereinbart werden — im Regelfall dürfte es sich um jeweils ein *Point*- oder *Polygon*-Objekt handeln.

Von der Fachapplikation wird dieses Feld 1:1 als Text abgelegt, weshalb sie das Format höchstens soweit interpretieren muss, um die Koordinaten im GUI anzuzeigen.

```
{
      "type": "Point",
      "coordinates": [2609190, 1226652]
}
```

Neben GeoJSON standen auch Well-known Text (WKT) und Well-known Binary (WKB) zur Diskussion, ausschlaggebend für den Entscheid zugunsten GeoJSON waren die folgenden Punkte:

- Mit GeoJSON können sowohl die umgebenden Nachrichten als auch die Geometriedaten als JSON übertragen werden; es müssen keine unterschiedlichen Parser eingesetzt werden.
- Gerade wenn nur Koordinatenzahlen angezeigt werden sollen braucht es auf Seiten der Fachapplikation nur sehr rudimentäre Kenntnisse des Formats. Da die CCC-Schnittstelle insbesondere zur Integration von Fachapplikationen ohne GIS-Funktionalität verwendet werden soll, passt dies sehr gut.
- Nachteil: GeoJSON unterstützt keine Koordinatensysteme und keine Kreisbogengeometrien.

3.8 notifyError

Zeigt an, dass bei der Verarbeitung Fehler aufgetreten sind und beschreibt diesen Fehler. Eine Antwortnachricht darf nicht sowohl ein ***Fehler! Verweisquelle konnte nicht gefunden werden.***- als auch ein *data*-Objekt enthalten. Sollten dennoch beide vorhanden sein muss das *data*-Feld ignoriert werden.

Ein ***Fehler! Verweisquelle konnte nicht gefunden werden.***-Objekt enthält einen numerischen *code*, der sich an die HTTP-Statuscodes anlehnt. Ausserdem enthält jedes ***Fehler! Verweisquelle konnte nicht gefunden werden.***-Objekt eine beschreibende *message*.

```
{
      "code": 461,
      "message": "First and last position in ring must be equal",
}
```

3.8.1 400 Ungültige Nachricht

Der Inhalt der Nachricht konnte nicht verarbeitet werden und es trifft keiner der anderen Gründe aus dem 400-er Bereich zu.

3.8.2 461 Ungültiges GeoJSON-Objekt

Das in der Nachricht angegebene GeoJSON-Objekt konnte nicht verarbeitet werden, der Inhalt von *data* entspricht nicht dem GeoJSON-Standard.

3.8.3 462 Geometrietyp wird nicht unterstützt

Der in einem GeoJSON-Feature angegebene Geometrietyp wird von dieser Anwendung nicht unterstützt.

3.8.4 463 GeoJSON-Typ wird nicht unterstützt

Der im GeoJSON-Objekt angegebene Objekttyp wird nicht unterstützt.

3.8.5 464 Falsches Referenzsystem

Die in der Geometrie eingegebenen Koordinatenzahlen scheinen sich auf ein anderes Referenzsystem als CH1903+/LV95 zu beziehen.

Erfolgt über die Anzahl Stellen

3.8.6 500 Interner Fehler

Die Nachricht konnte aufgrund eines nicht weiter spezifizierten Fehlers nicht bearbeitet werden.

3.8.7 503 Session nicht verfügbar

Für die Session besteht keine Verbindung zur Gegenstelle. Die Nachricht kann nicht zugestellt werden.

3.8.8 504 Session bereits verbunden

Für die im Feld *session* angegebene Session wurde bereits eine Verbindung registriert. Das alte Pairing der Verbindungen wird beibehalten.

3.8.9 505 Version wird nicht unterstützt

Das *apiVersion*-Feld enthält einen anderen Wert als "1.0".

3.8.10 506 Session Timeout

Der CCC-Server hat 60 Sekunden nach dem Empfang einer ***Fehler! Verweisquelle konnte nicht gefunden werden.***-Nachricht noch keine connectGis-Nachricht für die selbe *session* erhalten; beziehungsweise 60 Sekunden nach einer connectGis-Nachricht noch keine ***Fehler! Verweisquelle konnte nicht gefunden werden.***-Nachricht mit gleicher *session*. Nach dem Senden dieser Nachricht wird die Verbindung getrennt.

3.9 Beispiele

3.9.1 Nachricht zum Bearbeiten eines Polygons mit Attributen zur Anzeige im GIS

```
{
	"method": "editGeoObject",
	"context": {
		"zonen_num": CH493287040689
	},
	"data": {
		"type": "Polygon",
		"coordinates": [
			[
				[ 2642807.776, 1250834.890 ],
				[ 2642808.995, 1250833.309 ],
				[ 2642801.593, 1250815.236 ],
				[ 2642793.680, 1250799.885 ],
				[ 2642774.428, 1250810.681 ],
				[ 2642775.061, 1250811.930 ],
				[ 2642765.744, 1250816.665 ],
				[ 2642769.744, 1250821.588 ],
				[ 2642774.689, 1250827.672 ],
				[ 2642784.519, 1250840.170 ],
				[ 2642806.979, 1250835.924 ],
				[ 2642807.776, 1250834.890 ]
			]
		]
	}
}
```

4. OPERATIONEN

4.1 Handshake-Operationen

Die Handshake-Operationen dienen dem Aufbau der Verbindung zwischen den Clients mittels CCC-Server. Diese sind typischerweise für den Endbenutzer nicht direkt erkennbar. Beispielsweise klickt der Endbenutzer in der Fachapplikation auf den Button «Auf Karte zeigen». Sofern noch keine CCC-Verbindung zum Web GIS Client besteht, wird diese von der Fachapplikation mittels Handshake-Operationen initiiert (Für Endbenutzer nicht direkt erkennbar). Bei stehender Verbindung ruft die Fachapplikation die Payload-Operation «showGeoObject» auf.

Wichtiger Teil des Handshake ist das Öffnen des Web GIS Fensters durch die Fachapplikation. Mit dem Öffnen des Browsers mit der entsprechenden URL wird dem Web GIS Client die Session bekanntgemacht auf die er sich mittels «connectGis» verbinden soll. Siehe dazu das Sequenzdiagramm in Kapitel 5, Methoden createGeoObject() und navigateTo(Url, Session).

4.1.1 connectApp

Nach dem Aufbau der WebSocket-Verbindung zum CCC-Service muss die Fachanwendung diese Methode aufrufen, um die *session* bekanntzumachen.

Beispiel:

```
{
      "apiVersion": "1.0",
      "method": "connectApp",
      "session": "{E9C62508-025A-4A0F-B342-5A632282ABD8}",
      "clientName": "Axioma Mandant AfU"
}
```

Struktur des JSON-Aufrufes:

Attribut	Json-Typ	Nullable
apiVersion	String	N
method	String	N
session	String. Der String enthält die in Hex codierte UUID mit Klammern {}	N
clientName	String	N

4.1.2 connectGis

Nach dem Aufbau der WebSocket-Verbindung zum CCC-Service muss die WebGIS- Anwendung diese Methode aufrufen, um die *session* bekanntzumachen. Damit wird ausserdem deklariert, dass die WebGIS-Anwendung fertig initialisiert und bereit zum Empfang von CCC-Nachrichten ist.

Dazu muss die Session ID zuerst von der Fachapplikation an die WebGIS-Anwendung übergeben werden. Ein geeigneter Weg ist beispielsweise als URL-Parameter beim Laden der Web-Anwendung.

Beispiel:

```
{
	"apiVersion": "1.0",
	"method": "connectGis",
	"session": "{E9C62508-025A-4A0F-B342-5A632282ABD8}",
	"clientName": "Web GIS Client"
}
```

Struktur des JSON-Aufrufes:

Attribut	Json-Typ	Nullable
apiVersion	String	N
method	String	N
session	String. Der String enthält die in Hex codierte UUID mit Klammern {}	N
clientName	String	N

4.1.3 notifySessionReady

Sobald beide Seiten einer *session* bestehen (es wurden sowohl ***Fehler! Verweisquelle konnte nicht gefunden werden.*** als auch connectGis mit demselben *session*-Parameter aufgerufen), schickt der CCC-Service diese Nachricht an beide Applikationen um diese zu informieren, dass die jeweilige andere Applikation bereit zum Empfang von Befehlen ist. Daraufhin können beispielsweise in der Fachapplikation die entsprechenden GUI-Funktionen aktiviert werden.

```
{
	"apiVersion": "1.0",
	"method": "notifySessionReady"
}
```

Struktur des JSON-Aufrufes:

Attribut	Json-Typ	Nullable
apiVersion	String	N
method	String	N

4.2 Payload-Operationen

Die Payload-Operationen lösen in der Fachapplikation respektive im WebGIS die entprechenden und für den Endbenutzer direkt erkennbaren Operationen aus.

4.2.1 createGeoObject

Die Fachapplikation sendet diese Nachricht via CCC-Service an die WebGIS-Anwendung um ein neues GIS-Objekt zu erzeugen, beispielsweise einen Punkt zu setzen. In dieser Nachricht darf das *data*-Feld nicht gesetzt sein.

Beim Empfang zeigt die WebGIS-Anwendung die anwendungsspezifischen Werkzeuge zum Erstellen eines neuen GIS-Objektes für das Fachobjekt.

```
{
	"method": "createGeoObject",
	"context": {
		"afu_geschaeft": 3671951
	},
	"zoomTo": {
		"gemeinde": 2542
	}
}
```

Struktur des JSON-Aufrufes:

Attribut	Json-Typ	Nullable
method	String	N
context	Objekt. Inhalt des Objekts abhängig vom “Gespann” Fachapplikation – GIS Client	N
zoomTo	Objekt. Inhalt des Objekts abhängig vom “Gespann” Fachapplikation – GIS Client	J

Weiteres Beispiel:

```
{
	"method": "createGeoObject",
	"context": {
		"geschaeft_laufnr": "2017-820"
	},
	"zoomTo": [
		{
			"nbident": "SO0200002513",
			"nummer": 3765
		},
		{
			"gemeinde": 2542
		}
	]
}
```

4.2.2 editGeoObject

Die Fachapplikation sendet diese Nachricht via CCC-Service an die WebGIS-Anwendung, um ein GIS-Objekt zu ersetzen. Das *data*-Feld muss ein gültiges GeoJSON-Objekt enthalten.

Die WebGIS-Anwendung zeigt das erhaltene Objekt an und aktiviert die anwendungsspezifischen Werkzeuge zum Bearbeiten, Ersetzen oder Löschen der GIS-Objekte zum Fachobjekt.

```
{
	"method": "editGeoObject",
	"context": {
		"afu_geschaeft": 3671951
	},
	"data": {
		"type": "Point",
		"coordinates": [2609190, 1226652]
	}
}
```

Struktur des JSON-Aufrufes:

Attribut	Json-Typ	Nullable
method	String	N
context	Objekt. Inhalt des Objekts abhängig vom “Gespann” Fachapplikation – GIS Client	N
data	Objekt. [data] muss ein gültiges GeoJSON -Objekt sein	N

4.2.3 showGeoObject

Die Fachapplikation sendet diese Nachricht via CCC-Service an die WebGIS-Anwendung um ein GIS-Objekt anzuzeigen ohne die Werkzeuge zum Bearbeiten zu aktivieren. Der Inhalt dieser Nachricht entspricht der editGeoObject-Nachricht.

Je nach Anwendung und den Rechten des Benutzers kann es nützlich sein, direkt im WebGIS die Möglichkeit anzubieten, in den *Bearbeiten*-Modus zu wechseln.

```
{
	"method": "showGeoObject",
	"context": {
		"afu_geschaeft": 3671951
	},
	"data": {
		"type": "Point",
		"coordinates": [2609190, 1226652]
	}
}
```

Struktur des JSON-Aufrufes: Identisch zur Operation [editGeoObject]

4.2.4 cancelEditGeoObject

Die Fachapplikation sendet diese Nachricht via CCC-Service an die WebGIS-Anwendung um anzuzeigen, dass das betroffene Objekt nicht mehr bearbeitet werden darf. Beispielsweise, da die entsprechende *Bearbeiten*-Funktionalität in der Fachapplikation beendet wurde.

Falls die WebGIS-Applikation nicht am Bearbeiten des in context angegebenen Objektes ist, kann sie diese Nachricht verwerfen.

```
{
	"method": "cancelEditGeoObject",
	"context": {
		"afu_geschaeft": 3671951
	}
}
```

Struktur des JSON-Aufrufes:

Attribut	Json-Typ	Nullable
method	String	N
context	Objekt. Inhalt des Objekts abhängig vom “Gespann” Fachapplikation – GIS Client	N

4.2.5 notifyEditGeoObjectDone

Die WebGIS-Anwendung sendet diese Nachricht via CCC-Service an die Fachapplikation sobald der Benutzer die Bearbeitung abgeschlossen hat (z.B. im Fall BauGK einen Punkt gewählt und auf *Koordinaten übernehmen* geklickt hat). Die Nachricht enthält im *data*-Feld die von der Fachapplikation zu speichernden GIS-Daten. Falls *data* den JSON-Wert *null* hat bedeutet dies, dass ein etwaig vorhandenes GIS-Objekt zum Datensatz gelöscht werden soll.

Die WebGIS-Anwendung darf das Objekt nicht speichern, die Fachanwendung ist dafür verantwortlich, die GIS-Daten über die *WriteData*-Schnittstelle in den GIS-Cache zu bringen.

```
{
	"method": "notifyEditGeoObjectDone",
	"context": {
		"afu_geschaeft": 3671951
	},
	"data": {
		"type": "Point",
		"coordinates": [2609190, 1226652]
	}
}
```

Struktur des JSON-Aufrufes:

Attribut	Json-Typ	Nullable
method	String	N
context	Objekt. Inhalt des Objekts abhängig vom “Gespann” Fachapplikation – GIS Client	N
data	Objekt. [data] muss ein gültiges JSON-Objekt sein oder auf den JSON-Wert null gesetzt sein.	J

Beispiel bei Löschung des Geo-Objektes :

```
{
	"method": "notifyEditGeoObjectDone",
	"context": {
		"laufnr": "2017-820"
	},
	"data": null
}
```

4.2.5.1 Sequenzdiagramm für «notifyEditGeoObjectDone» Operation

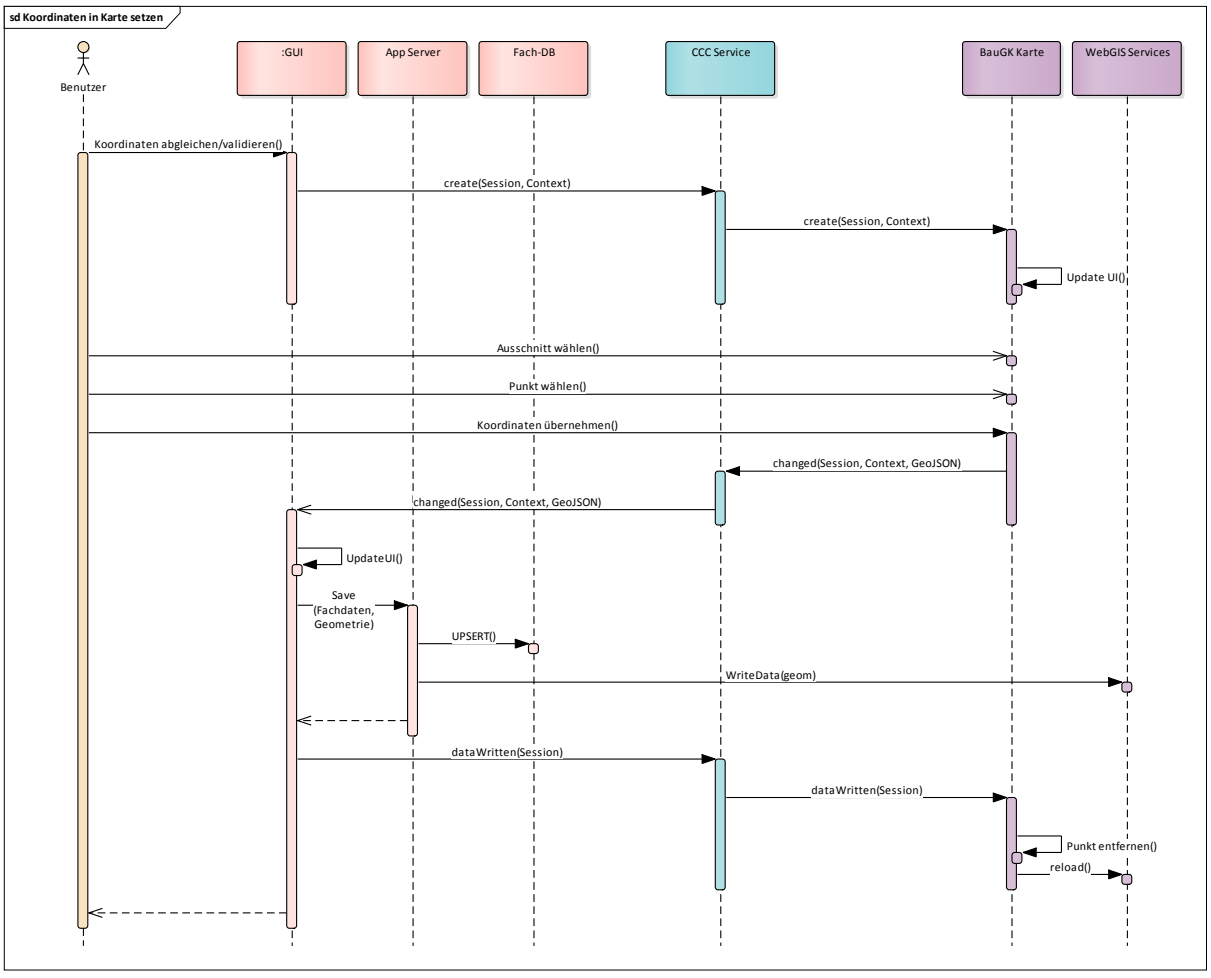

Abbildung 2: Sequenzdiagramm für das Auswählen von Koordinaten in einer Karte

Das Sequenzdiagramm in Abbildung 2 zeigt wie Koordinaten gewählt werden und dass anschliessend die Fachapplikation dafür zuständig ist, diese Daten zu speichern und per *WriteData* an das WebGIS zu übergeben.

4.2.6 notifyObjectUpdated

Die Fachapplikation sendet diese Nachricht an die WebGIS-Anwendung um dieser mitzuteilen, dass der GIS-Cache via *WriteData*-Schnittstelle geändert wurde und die WebGIS-Anwendung den

Datensatz neu laden muss. Die zur Identifikation nutzbaren Attribute werden wie bei notifyGeoObjectSelected im Feld *properties* abgelegt.

Die ***Fehler! Verweisquelle konnte nicht gefunden werden.***-Nachricht wird sowohl bei Änderungen an der Geometrie als auch bei Attributänderungen ausgelöst.

```
{
	"method": "notifyObjectUpdated",
	"properties": [{
		"laufnr": "2017-820",
		"grundbuch": "Trimbach;",
		"geschaeftseigner": "Verfahrenskoordination, Information",
		"gesuchsart": "Baugesuch"
	}]
}
```

Struktur des JSON-Aufrufes:

Attribut	Json-Typ	Nullable
method	String	N
properties	Array von Objekten	N

4.2.7 notifyGeoObjectSelected

Die WebGIS-Anwendung sendet diese Nachricht via CCC-Service an die Fachapplikation sobald der Benutzer ein oder mehrere Objekte aus dem GIS-Cache auf der Karte anklickt/selektiert. Im Gegensatz zur Operation notifyEditGeoObjectDone werden die Attribute der context_list aus dem GIS-Cache gelesen. Die Attribute werden von der Fachapplikation via *WriteData* gepflegt (siehe Abschnitt 0, notifyEditGeoObjectDone), und können deshalb auch von dieser zur Identifikation der betroffenen Fachobjekte verwendet werden. Es sollen primär die zur Identifikation geeigneten Attribute übergeben werden.

Typischerweise werden die Objekte durch n Name-Wert-Paar repräsentiert. Das genaue Format der Objekte kann jedoch für jede Anwendung festgelegt werden.

```
{
	"method": "notifyGeoObjectSelected",
	"context_list": [{
			"bfs_num": 231
			"parz_num": 1951
		},
		{
			"bfs_num": 231
			"parz_num": 2634
		}
	]
}
```

Ein Aufruf mit null als properies-Wert gibt zum Ausdruck, dass die Selektion aufgehoben wurde.

```
{
      "method": "notifyGeoObjectSelected",
      "context_list": null
}
```

Struktur des JSON-Aufrufes:

Attribut	Json-Typ	Nullable
method	String	N
context_list	Array von Objekten.	J

4.3 NotifyError-Operation

Mit der notifyError-Operation teilt ein Endpunkt dem entsprechenden client mit, dass ein Fehler bezüglich der Verarbeitung der Operation vorliegt.

Erzeuger von Fehlermeldungen sind bei Payload-Operationen die Fachapplikation respektive der Web GIS Client. Die Fehler werden durch den CCC-Server dem anderen Client zugestellt.

Bei Protokollfehlern, Fehlern bei den Handshake-Operationen und Fehlern aufgrund ungültigem Zustand der CCC-Session erzeugt der CCC-Server den Fehler und schickt diesen an den aufrufenden Client.

Beispiel für einen NotifyError aufgrund einer showGeoObject-Operation:

```
{
      "method": "notifyError",
      "code": 464,
      "message": "Coordinate must have seven digits before decimal point",
      "userData": {
                  "afu_geschaeft": 3671951
                  "coordinates": [609190, 226652]
      },
      "nativeCode": "Err-165",
      "technicalDetails": "java.lang.IllegalArgumentException:
        at example.common.TestTry.execute(TestTry.java:17)
        at example.common.TestTry.main(TestTry.java:11)"
}
```

Struktur des JSON-Aufrufes:

Attribut	Json-Typ	Nullable	Bemerkung
method	String	N	
code	Integer	N	Fehlercode – siehe Kapitel **Fehler! Verweisquelle konnte nicht gefunden werden.**

message	String	N	Fehlermeldung – siehe Kapitel **Fehler! Verweisquelle konnte nicht gefunden werden.**
userData	JSON-Objekt	J	Information zu Objekt / Objektattributen, welche den Fehler ausgelöst haben
nativeCode	String	J	Codierte Kennung des Fehlers im entsprechenden Endpunkt
technicalDetails	String	J	Ergänzende Informationen zum aufgetretenen Fehler. z.B: Stacktrace

5. CLIENT-CLIENT CONTEXT SERVICE

Der CCC-Service dient als zentraler Punkt über den die beiden Clients (Fachapplikation und WebGIS) ihre Nachrichten austauschen. Der Service verwaltet die beiden zu einer *session* gehörenden WebSocket-Verbindungen und leitet nach der Initialisierungsphase Nachrichten an den jeweils anderen Kommunikationspartner weiter; die *session* ist ab dann implizit.

Die zweite Connect-Nachricht einer Session (***Fehler! Verweisquelle konnte nicht gefunden werden.*** nach connectGis bzw. connectGis nach **Fehler! Verweisquelle konnte nicht gefunden werden.**) muss innerhalb von 60 Sekunden (konfigurierbar) nach der ersten beim CCC-Server eintreffen. Bei Überschreiten der Zeitspanne muss der Server bei der nächstfolgenden Operation mit NotifyError 506 antworten (Session timeout).

Falls auf einer WebSocket-Verbindung nach der ersten **Fehler! Verweisquelle konnte nicht gefunden werden.**- oder connectGis-Nachricht eine weitere ***Fehler! Verweisquelle konnte nicht gefunden werden.***- oder connectGis-Nachricht empfangen wird muss diese ignoriert und mit *Fehler 504 Session bereits verbunden* beantwortet werden um zu vermeiden, dass Sessions übernommen werden können.

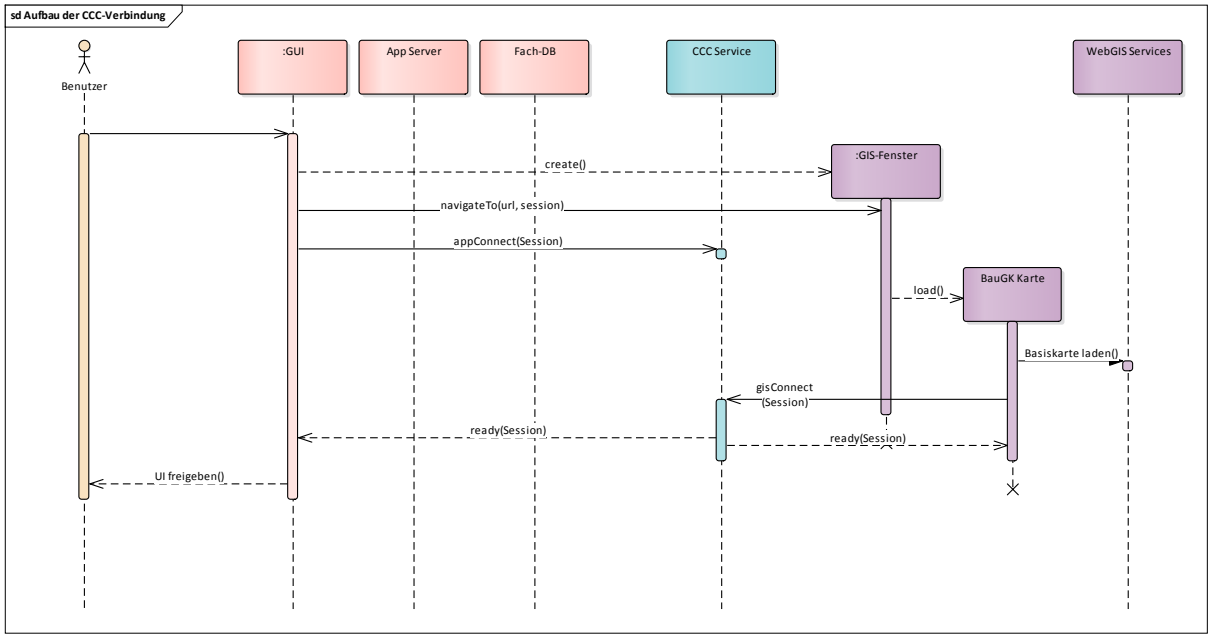

Abbildung 3: Sequenzdiagramm für den Verbindungsaufbau. Das GIS-Fenster ist direkt in die Fachapplikation integriert (Web View/Web Control), weshalb die Schritte createGeoObject() *und* navigateTo() *getrennt sind. Falls das WebGIS im Standardbrowser geöffnet werden soll, erfolgen diese beiden Schritte in einem Rutsch:* CreateGeoObjectProcess(url, session)*.*

Sobald eine der beiden WebSocket Verbindungen geschlossen wird (*Close*-Frame oder falls ein Verbindungsfehler auftritt) schliesst der CCC-Service auch das Gegenstück.

6. SICHERHEIT

- Für die Verwaltung der Berechtigungen zum Anzeigen und Schreiben der Fachdaten (inkl. übertragenen GIS-Objekte) ist die Fachapplikation zuständig.
- Die Anzeige der Kartenobjekte in der WebGIS-Anwendung wird über die Benutzer- und Rechteverwaltung der Geodateninfrastruktur gesteuert.
- Der WebSocket Datenverkehr wird per TLS verschlüsselt (*WebSocket Secure*-Protokoll).
- Alle über die CCC-Schnittstelle eingehenden Nachrichten müssen als potenziell gefährlich eingestuft werden da sie von einem Client erstellt werden und modifiziert werden können. Eingehende Daten müssen, wie alle Benutzereingaben, vor der Verarbeitung/Anzeige geprüft und ggf. escaped werden bevor sie weiterverwendet werden.
- Die Weitergabe der GIS-Daten von der Fachapplikation an das GIS (Cache) erfolgt über die *WriteData*-Schnittstelle des GIS. Da die Datenhoheit für diesen Datensatz bei der Fachapplikation liegt, kann hierfür ein entsprechender Applikationsbenutzer eingesetzt werden. Es ist nicht notwendig, diese Operation als der Fachbenutzer auszuführen.

Der CCC-Service kann dazu verwendet werden vor Beenden des Sessionaufbaus beliebige Nachrichten an den jeweils zuerst aktiven Verbindungspartner zu schicken sofern der Wert von *session* bekannt ist. Eine Möglichkeit, diesen Angriffspunkt zu eliminieren ist, den Aufbau einer Session nur für ein und denselben Benutzer zuzulassen.
Siehe https://devcenter.heroku.com/articles/websocket-security#authentication-authorization zu weiterführenden Informationen zu WebSocket-Authorisierung.

Dennoch wurde in der vorliegenden Version des Protokolls darauf verzichtet, die Session an den Benutzer zu binden, da unklar ist, wie beide Applikationen ihre jeweiligen Sessions so weitergeben können, dass dies

a) transparent für den Benutzer ist,

b) der CCC-Server prüfen kann, dass die beiden Verbindungen zum selben Benutzer gehören, und

c) der Aufwand für die Umsetzung dieser Prüfung in einem vernünftigen Verhältnis zum Bedrohungsmodell steht, insbesondere da die Anwendung nur im Intranet genutzt wird.

Wenn bei folgenden Integrationen eine Verwendung ausserhalb des Intranet angestrebt wird, muss die Sicherheit erneut analysiert werden. Voraussichtlich ist es hinreichend in einer entsprechend angepassten Version zu verlangen, dass derselbe Benutzer beide WebSocket-Verbindungen der CCC-Session authentisiert.

Ausserdem wurde das Risiko eines erfolgreichen Angriffs minimiert, indem pro Richtung nur jeweils eine WebSocket-Verbindung die selbe *session* verwenden darf; nachfolgende ***Fehler! Verweisquelle konnte nicht gefunden werden.***- bzw. connectGis-Nachrichten mit gleicher *session* müssen ignoriert und, falls auf anderen Verbindungen, mit Fehler *504 Session bereits verbunden* beantwortet werden.

Ein weiteres Angriffsszenario besteht darin, sich beim CCC-Service als Fachapplikation zu registrieren (***Fehler! Verweisquelle konnte nicht gefunden werden.***) und anschliessend den Benutzer dazu zu bringen, die WebGIS-Applikation mit gleichem Context zu öffnen. Dieser Schritt kann auch per Link in einer E-Mail-Nachricht erfolgen. Daraufhin werden für alle vom Benutzer angeklickten Objekte aus dem GIS-Cache die Attribute an den Angreifer geschickt; Daten können allerdings weder aktiv abgefragt noch verändert werden.

Das Risiko dieses Angriffs wird verringert indem der CCC-Service die Verbindung 60 Sekunden nach der ersten Connect-Nachricht mit Fehler *506* abbricht sofern die Verbindung nicht durch die Connect-Nachricht der Gegenstelle beantwortet wurde. Die möglichen Auswirkungen werden minimiert, indem bei der notifyGeoObjectSelected-Nachricht nur die zur Identifikation des Objekts notwendigen Attribute mitgeschickt werden.