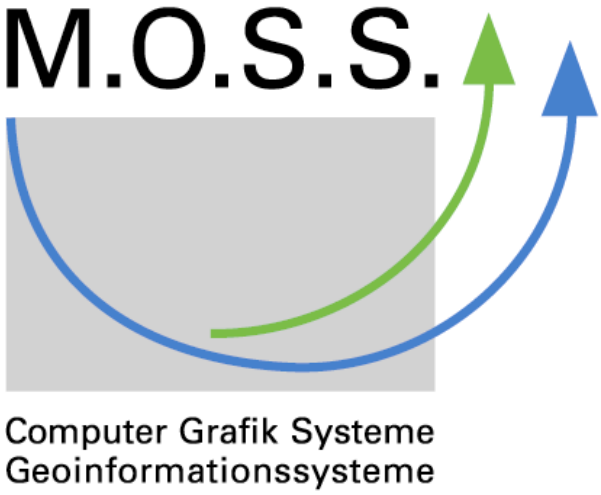
WEGA-GDM
GeoServer

WEGA-GDM Version 2.4



Copyright© 1997-2002

M.O.S.S. Computer Grafik Systeme GmbH

Postanschrift: Hohenbrunner Weg 13, 82024 Taufkirchen

Telefon: (089) 666 75 100 Fax: 666 75 180

Geschäftsführer: Dipl.-Math. Hans Braun

HR B: 82009 Amtsgericht-Registergericht-München

Alle Rechte vorbehalten

Autoren: K-H. Bussian

Datum: 12/09/2002

Wir bieten unseren Kunden nicht nur Software und Handbücher, sondern auch Schulungen und Beratungen zu folgenden Themenbereichen an:

- zum vorliegenden Produkt
- zu RoSy und dessen Umfeld
- zu KANDIS und dessen Umfeld
- zu SICAD-ROSY und SICAD
- zu den WEGA-Produkten und dessen Umfeld
- zu den Programmierwerkzeugen Tcl/Tk und EASI

Die von M.O.S.S. angebotenen Schulungen werden entweder in unserem Münchner Hauptsitz, in den Geschäftsstellen Dresden, Hamburg und Essen oder direkt beim Kunden durchgeführt.

Informationen zu unserem Schulungsangebot bekommen Sie per Fax:

+49-89-66675-180

Oder Sie schreiben an:

M.O.S.S. Computer Grafik Systeme GmbH

Abteilung Vertrieb

Hohenbrunner Weg 13

D-82024 Taufkirchen

Das aktuelle Schulungsangebot können Sie auch direkt abrufen über:

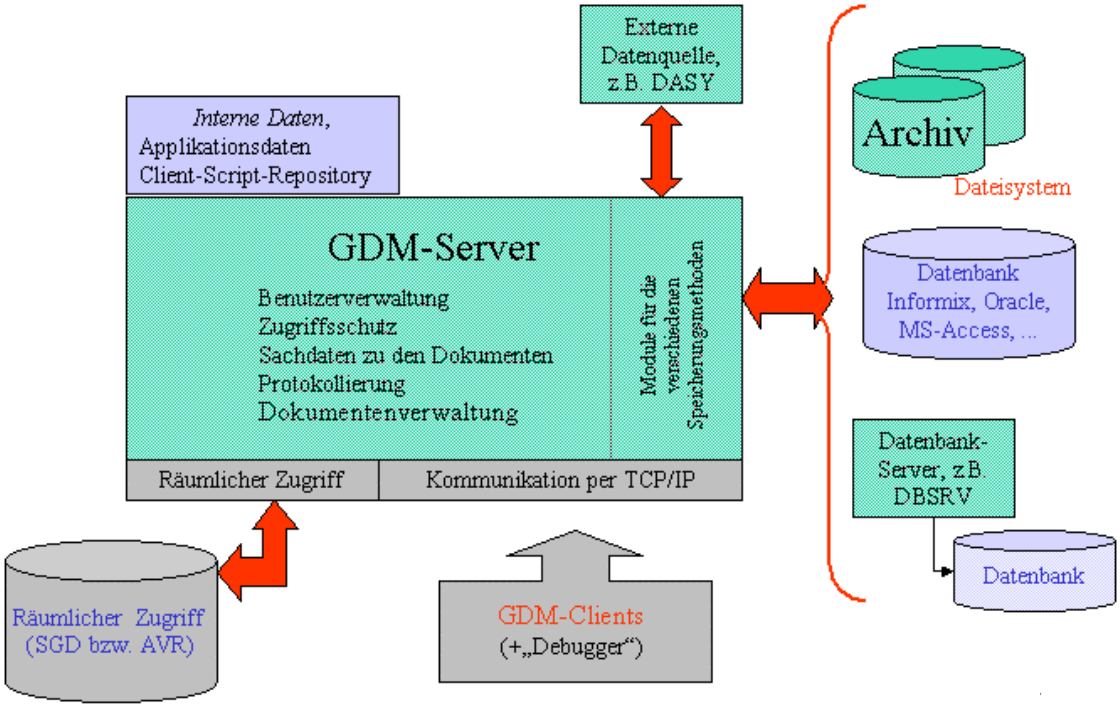
http://www.moss.de

Zur Anforderung des aktuellen Schulungsangebots stehen Ihnen auf der letzten Seite dieses Handbuchs Fax-Formulare zur Verfügung.

Fax-Formulare für kritische Anmerkungen oder Anregungen zum vorliegenden Handbuch sind ebenfalls an dieser Stelle zu finden.

Inhaltsverzeichnis

Architektur des GeoServers:



UNIX-Environmentvariablen für Wega-GDM

Alle Environmentvariablen des Products Wega beginnen mit dem Vorspann "WEGA_". Es muß zumindest die Variable

WEGA_GDMPATH auf das Verzeichnis zeigen, wo die Wega-GDM Software sich befindet.

andernfalls wird diese Variable beim beim Starten von Unix-Skripten auf den Defaultwert "`/moss_gdm`" gesetzt. Aus dieser Variablen leitet sich der Pfad zum Server per

```
$WEGA_GDMPATH/server
```

ab.

Die weiteren Variablen sind:

WEGA_PRODUCTNAME Bezeichnung des WEGA-Produkts (WEGA-

	GDM)
WEGA_GDMRELEASE	Aktuelle Release-Version des Servers
WEGA_GDMPARAMFILE	Name der Parameterdatei. Der Name muß entweder als absoluter Pfad oder relativ zum Server-Verzeichnis angegeben sein.
WEGA_GDMSGDPATH	Pfad zu den SGD-Dateien. Hier werden defaultmäßig auch die Log-Dateien angelegt.
WEGA_GDMSGDNAME	Name der SGD-GeoBasis-Datei
WEGA_GDMSGDFILENAME	Alternativ kann hier der vollständige Pfad zu der SGD-Datei angegeben werden.
WEGA_GDMDBNAME	Name der Datenbank mit den GDM-Informationen und Blobs
WEGA_GDMFILEPATH	Pfad zu Dokument-Dateien im Verzeichnissystem (Cache).
WEGA_GDMDATPATH	Pfad zum Dateipool. Von hier werden alle Unterverzeichnisse des zur Suche von Dokumenten abgeleitet.
WEGA_GDMAPPLNAME	Name der aktuellen GDM-Serverapplikation enthält installationsspezifische Skripte bzw. Dateien.
WEGA_GDMAPPLPATH	Pfad zu applikationsspezifischen Dateien
WEGA_GDMSRVPORT	Portnummer, die der GDM-Server benutzen soll (kann via Parameterdatei geändert werden).
WEGA_GDMSRVHOST	Rechnername auf dem der GeoServer läuft (default: aktueller Rechner)
WEGA_GDMSERVLETS	Liste der Servlets, die der Server beim Start laden und evtll. ausführen soll.
WEGA_GDMLOGFILE	Name der Protokollierungsdatei
WEGA_GDMICONS	Verzeichnis mit Icons, etc. (Default `$WEGA_GDMPATH/images`)
WEGA_GDMPID	Prozeßidentifikation des GeoServers
WEGA_GDMLICENSE	der Lizensierungscode.
WEGA_GDMTEMPDIR	Verzeichnis zum Speichern von Zwischendateien.

WEGA_GDMZVFSNAME vollständiger Dateiname für die GDM-Server-Runtime-Bibliothek (*BTL*=Binary Tcl Library)

zur **Datensicherung** (Backup) sind die folgenden Variablen definiert:

WEGA_GDMTAPEDEV Bandheinheit auf der die Datensicherung durchgeführt wird.

WEGA_GDMTAPECHECK Uhrzeit (Stunde), ab der getestet wird, ob ein Band eingelegt ist (0-23).

WEGA_GDMTAPEPERFORM Uhrzeit (Stunde), ab der die Datensicherung auf Band erfolgen soll (0-23).

WEGA_GDMTAPEPEREWIND Flag ob das Sicherungsband zurückgespult werden soll, bevor die Datensicherung beginnt.

WEGA_GDMUSETAPE Flag ob Sicherung auf Band geschrieben werden soll oder nur Sicherung auf Platte. Sinnvoll, wenn Datensicherung der Platten standardmäßig vorhanden ist

WEGA_GDMTAPEMAILTO Liste von Mailaddressen, die die Backup-Logs oder Fehlermeldungen erhalten sollen.

GeoServer-Starten/Stoppen

Der GeoServer wird unter ***Unix***-Betriebssystemen mit Hilfe der Skripte ***StartServer***, ***ShutServer*** und ***PingServer*** gestartet, gestoppt bzw. auf Antwort geprüft. Es muß dazu entweder die Environmentvariable

WEGA_GDMPATH auf das Verzeichnis zeigen, wo die Wega-GDM Software sich befindet.

oder es muß ein symbolischer Link auf das Vater-Verzeichnis vorher erstellt worden sein, z.B.:

```
ln -s /disk2/users/gdm   /moss_gdm
```

die Server-Software befindet sich dann im Verzeichnis

```
/moss_gdm/server
```

Üblicherweise wird das Environment für WEGA-GDM wird mit Hilfe des Startskriptes "***wega.enb_ksh"*** aktiviert (dieses Skript wird bei der Installation erstellt bzw. angepaßt).

WinNT:

Unter WinNT wird der GeoServer mit der Batchdatei "***server.bat***" aktiviert, wobei als 1. Parameter die geforderte Aktion, nämlich

start	Starten des GeoServers inkl. eventuell notwendiger DB-Prozesse/-Server
stop	Shutdown (Runterfahren) des GeoServers
ping	Prüfen ob Server aktiv
client	Starten des GDM-Clients (normalerweise ist dies per Explorer oder Startmenü einfacher)
monitor	Starten des Debugging Monitors um den Geo Server debuggen zu können
recover	Recovery-Aktion durchführen, s. Beschreibung unten
setup	nur einmalig aufzurufen, um den Installationspfad für Hilfsprogramme etc. einzutragen.

angibt und als 2. Parameter die GeoServer-Steuerdatei anzugeben ist. Je nach Installation, kann die entsprechende Steuerdatei bereits in der Batchdatei als Default hinterlegt sein, dann ist nur die Angabe des 1. Parameters notwendig.

Je nach Installation kann der GeoServer unter WinNT auch per Systemdienst gesteuert werden, so daß der Server auch dann weiterläuft, wenn der Benutzer sich ausloggt, bzw. kein Benutzer eingeloggt ist. Der Systemdient hat auch den Vorteil, daß der Server beim Booten hochgefahren und bei Shutdown des Rechners beendet wird.

Um den Systemdienst einzurichten, ist die Datei GDMServer.ini entsprechend zu editieren und anschließend ist der Dienst zu aktivieren per Aufruf des Programms GDMServer:

```
GDMServer -i    (Installiert den Dienst GDMServer)
GDMServer -s    (startet den Dienst und damit den GeoServer)
```

Die Servereinstellungen sind in einer Parameterdatei (der Server-Steuerdatei) zu hinterlegen. Diese Datei kann dann bei den Aufrufen angegeben werden oder per Environmentvariable

WEGA_GDMPARAMFILE

definiert werden.

Unix:
Es wird der Server unter Unix gestartet mit

```
StartServer
```

die Initialisierung des Servers läuft im Hintergrund, der Server wird mit Hilfe des Unix-spezifischen "at"-Befehls gestartet. Man muß daher mindestens eine Minute warten bevor der Server wirklich aktiv wird. Um zu prüfen ob der Server aktiv ist, kann das Skript

```
PingServer
```

benutzt werden. Es sollte dann eine Antwort wie "*Server alive*" erscheinen. Das Skript kann nur auf dem Rechner aktiviert werden, auf dem der Server läuft. Um von einem anderen Unix-Rechner dieses Skript zu aktivieren, ist die Option "-remote" anzugeben, z.B.

```
PingServer -remote
```

(es muß aber trotzdem ein Teil der Serversoftware auf dem Rechner installiert sein, z.B. Tcl).

Der Server kann abgeschaltet werden mit Hilfe des Skriptes

```
ShutServer
```

Der Shutdown funktioniert nur dann, wenn der Server nicht gerade "hängt", dieser Effekt tritt auf, wenn ein weiterer Server auf dem gleichen Port aktiviert wird (zumindest tritt dies unter HP-UX) auf. Um von einem anderen Rechner den Server abzuschalten, kann man die Option "`-remote`" benutzen.

Zum Debuggen steht ein Monitorprogramm zur Verfügung. Dieses Programm wird mit

```
StartMonitor
```

aktiviert. Dieses Programm sollte nur in Ausnahmefällen aktiviert werden (dieses Programm kann mit der Option "-remote" auch von einem anderen Rechner aktiviert werden, jedoch muß dann ein Teil der Serversoftware installiert sein).

Als weitere Hilfe zum Testen/Debuggen, steht das Skript

```
QueryServer ?-remote? ?-setup?
```

zur Verfügung. Es kann die Zustandsinformationen eines Servers abfragen, bzw. die Environmenteinstellungen abfragen.

Unter Unix benutzen alle Serverskripte, wie StartServer, PingServer, ShutServer usw. das Basisskript

```
PidServer [<paramFile>]
```

um festzustellen, ob der Server bereits läuft.

> **ACHTUNG**:
> Derzeit wird ein Shutdown des Rechners unter Unix (zum Abschalten der CPU) noch nicht an dem GeoServer weitergereicht, es besteht dann die Gefahr, daß durch einen unkontrollierten Abbruch des GeoServers die GeoBase (SGD-Datei) zerstört wird! Es muß daher sichergestellt werden, daß beim Shutdown des Rechners der GeoServer nicht mehr aktiv ist!

Backup Server-Daten

Je nach Sicherungskonzept sind die Daten des Servers zu sichern. Entweder wird eine Bandeinheit dazu benutzt nur den GeoServer Datenbereich zu sichern, oder es wird die Platte regelmäßig gesichert und so auch die GeoServer-Daten. Defaultmäßig wird unter Unix das Skript

```
BackupServer
```

bereitgestellt. Dieses Skript sollte per "cron"-Befehl zu der gewünschten Uhrzeit starten und die Sicherung der Daten durchführen. Es wird hierbei die Datenbank in ein Temporärverzeichnis entladen und dann dies zusammen mit allen anderen Dateien auf Band gespielt. Während des Backups wird der GeoServer, sofern aktiv, heruntergefahren und anschließend wieder aktiviert. Die Datenbank wird bei einer Informix-Datenbank per "dbexport" auf ein Temporärverzeichnis geschrieben und dieser Bereich wird zusammen mit den Server-Dokumenten/Dateien gesichert.

Recovery von Backup der Server-Daten

Soll eine Wiederherstellung von Daten anhand eines Server-Daten Backup-Bandes erfolgen, sind die folgenden Schritte durchzuführen:

- GeoServer herunterfahren, falls GeoServer aktiv (Dies muß erfolgen, da sonst der GeoServer vom BackupServer-Skript runtergefahren und anschließend wieder gestartet wird)

```
ShutServer
```

- Sicherung des aktuellen Standes durch Aktivierung des Skriptes

```
BackupServer  -now
```

- Daten auf Band sichern, so daß der letzte Stand wiederhergestellt werden kann.

- "alte" Daten von Recovery-Band zurückschreiben. Es sollten dabei die Dokument- und die Datenbank-Export-Verzeichnisse vom Band gelesen werden. Welche Verzeichnisse hierbei zu entladen sind, kann mit Hilfe des Informationsskriptes

```
QueryServer -setup
```

  bestimmt werden, denn hier wird eine Liste der Einstellungen des Servers herausgegeben. Die Datenbanktabelle werden unter dem Verzeichnis ausgegeben, das per

```
WEGA_GDMTEMPDIR
```

  referenziert wird. Die Dokumente werden unter dem Verzeichnis ausgegeben, das per

```
WEGA_GDMDATPATH
```

  bzw.

```
WEGA_GDMFILEPATH
```

  angegeben wird.

- Wurde keine Änderung an der Datenbankstruktur seit der eingespielten Datensicherung durchgeführt, kann der Datenbankbestand unter Informix per "dbimport" in die Datenbank geladen werden. Es werden hierbei die Daten, die in dem Temporärverzeichnis unter Environmentvariablen

```
WEGA_GDMTEMPDIR
```

  definiert sind, geladen. Man kann hier auch gezielt einzelne Datenbanktabellen wiederherstellen indem nur die entsprechenden Ladedateien in die Datenbank importiert werden.

- Nach erfolgreichem Laden der Datenbanktabellen, ist die GeoBase-Datei zu re-initialisieren. Dazu muß ein Recovery auf die GeoBase-Datei durchgeführt werden:

```
Recover
```

  Es entsteht eine neue SGD-Datei im lokalen Verzeichnis. Nach Abarbeitung sollten die Meldungen analysiert werden und erst danach ist die neue SGD-

Datei in das Verzeichnis zu kopieren, wo die "alte" SGD-Datei sich befindet. Der Pfad zu dieser Datei ist per

```
WEGA_GDMSGDPATH/$WEGA_GDMSGDNAME
```

bzw.

```
WEGA_GDMSGDFILENAME
```

festgelegt.

- Nun kann man den GeoServer wieder aktivieren.

  ```
  StartServer
  ```

  Man sollte jedoch sich die Log-Datei ansehen. Diese Datei befindet sich dort, wo die SGD-Datei gespeichert wird, jedoch mit der Extension ".log". Sollten Probleme auftauchen, wird dies in der Log-Datei vermerkt.

- Sind die Meldungen in der Log-Datei in Ordnung, dann kann der GeoServer wieder freigegeben werden, d.h. GDM-Clients können wieder zugreifen.

Recovery GeoBase-Datei

Wird der Server unkontrolliert abgebrochen, dann kann es sein, daß die GeoBase (die SGD-Datei) nicht korrekt geschlossen wurde und beim nächsten ServerStart die Meldung

```
"Fehler in Freispeicherkette..
```

auftritt, wenn ein neues Dokument eingerichtet wird oder im Auskunftsmodus Dokumente gesucht werden. In diesem Fall kann die GeoBasis anhand der Datenbankinformationen rekonstruiert werden (sofern nicht Einträge per Laderoutine nur in der GeoBase vorgenommen wurden)

Es müssen dazu jedoch die sogenannten Crossreferenztabellen in der DB vorhanden sein und Dokumentinformationen dort eingetragen sein.

Es wird vorausgesetzt, daß die SGD-Datei in einem ursprünglichen "sauberen" Zustand vorliegt und diese Ursprungsdatei in der Einstellung "sgdMasterFile" des GeoServers definiert ist.

Beim Recovery-Vorgang werden nur die Dokumenttypen bearbeitet, die in der Typendefinition 'enabled' sind, alle anderen Einträge werden nach einer Fehlermeldung übergangen. Alle Dokumenteinträge, zu denen kein Dokument mehr vorhanden ist (z.B. gelöscht), werden ebenfalls übergangen.

Um eine Wiederherstellung der GeoBase zu aktivieren, ist zuerst einmal der GeoServer zu deaktivieren (ShutServer) und anschließend das Skript

```
Recover
```

zu aktivieren. Unter WinNT wird ein Recovery mit der Batchdatei "server.bat" aktiviert, wobei als 1. Parameter die geforderte Aktion, nämlich

```
server recover ...
```

anzugeben ist.

Es wird dann eine neue SGD-Datei im lokalen Verzeichnis erzeugt. Nach Abarbeitung sollten die Meldungen analysiert werden und erst danach ist die neue SGD-Datei in das Verzeichnis zu kopieren, wo die "alte" beschädigte SGD-Datei sich befindet. Anschließend kann der GeoServer wieder aktiviert werden.

Meldungen des GeoServers an den Client

Der GeoServer speichert zu jedem Client ein Information in welcher Sprache der Client Meldungen (insbesondere Fehlermeldungen) erhalten möchte. Der GeoServer holt sich Meldungen anhand von einheitlichen Meldungsnamen und der Spracheinstellung des Clients und substituiert anschließend Platzhalter für Parameter durch die aktuellen Daten für die Meldung. Das Ergebnis wird an den Client geliefert. Die Meldungen werden anschließend in einen internen Cache übernommen.

Die eigentlichen Meldungstexte sind in der Datenbank gespeichert. Um eine Datenbank mit den Meldungstexten zu initialisieren bzw. neue Texte zu hinzuzufügen, steht die Prozedur

```
dbloaderrmsg.tcl
```

zur Verfügung. Die Prozedur besitzt folgende Optionen

-dbconnect info	Angabe der DB-Verbindung, die benutzt werden soll. Hier sind die gleichen Angaben zu machen, wie bei dbConnect im GeoState-Zustandsfeld
-cretables	Falls die DB-Tabellen für die Fehlermeldungstext noch nicht vorhanden sind, dann werden sie erzeugt.
-log file	Umlenken der Ausgaben in die angegebene Datei, sonst Ausgabe auf `stdout`.
-msgnamtable tab	Anstelle des Defaultnamens ist die hier angegebene Namenstabelle zu benutzen. Hier werden Meldungsnamen (msgID) hinterlegt.
-msgstrtable tab	Anstelle des Defaultnamens ist die hier angegebene Meldungstexttabelle zu benutzen. Hier werden die

	Texte in den verschiedenen Sprachen hinterlegt.
-verbose	detaillierte Ausgabe der Aktionen
-infile msgfile	Diese Option muß mindestens angegeben werden. Es wird die hier angegebene Meldungsdatei gelesen und die hier angegebenen Texte in die Datenbank geladen

Die Meldungsdatei enthält alle Fehlermeldungen in den verschiedenen unterstützten Sprachen. Die Meldungsdatei kann eine Zuordnung von Meldungsname zu Meldungsnummer enthalten, ist diese nicht vorhanden, dann wird die Nummer zu einem Meldungsnamen von der Datenbank abgeleitet.

Für jede Sprache ist ein Textblock bereitzustellen, in der Form:

```
Sprache1 {
 Meldungsname AnzahlArgument "Meldungstext in Sprache1"
 ...
 ...
}
Sprache2 {
 Meldungsname AnzahlArgument "Meldungstext in Sprache2"
 ...
 ...
}
```

Um eine Übersetzung zu erhalten, muß jeder Meldungsname einem zugeordneten Text in den einzelnen Sprachblöcken besitzen.

Es können in den Meldungstexten Parameter verwendet werden. Die Anzahl der benötigten Parameter ist anzugeben. Ein Parameter wird im Text mit "%<nr>" referenziert, wobei <nr> die laufende Nummer des Parameters in der Aufrufliste darstellt. Es können maximal 9 Parameter angegeben bzw. referenziert werden. Werden beim Aufruf weniger Parameter geliefert, als benötigt, dann werden fehlende Parameter durch einen Leerstring ersetzt. Es können Parameter mehrmals und in unterschiedlicher Reihenfolge angegeben werden. Z.B.

```
german {
   MAXCLNTEXCD 1  "Maximale Anzahl von Clientverbindungen '%1' überschritten"
   UNKDOKTYP   1  "Unbekannter Dokumententyp '%1'"
   TYPDISABLED 1  "Der Dokumententyp '%1' ist derzeit deaktiviert"
   UNZIPDOKID  2  "Umsetzungsfehler dekomprimieren DokID: %1,%2"
}
english {
   MAXCLNTEXCD 1  "Maximum number (%1) of allowed client connects exceeded"
   UNKDOKTYP   1  "Unknown document type '%1'"
   TYPDISABLED 1  "Document type '%1' currently disabled"
   UNZIPDOKID  2  "Can't build unzipped dokID: %1,%2"
}
```

Zustandsinformationen des GeoServers

Der GeoServer speichert alle internen Informationen in den Zustandsarrays "geoState" und "geoClient", diese Arrays sollten nicht unkontrolliert, z.B. beim Debuggen mit Hilfe des GeoMonitors, geändert werden.

geoState-Informationen

Es werden im Array "`geoState`" alle allgemeinen Angaben hinterlegt. Viele dieser Angaben können mit Hilfe der installationsspezifischen Parameterdatei, die der Server beim Start einliest, gesetzt werden:

Komponente	Bedeutung
relVersion	Releaseversion des Servers in der Form *r.s.p* r = Majorrelease, s = Subversion, p = Patchlevel
protocolID	Server/Client-Protokollversion, wenn das Protokoll sich ändert, dann muß die Versionsnummer angepaßt werden.
licenseOwner	Bezeichnung des Lizenznehmers (z.B. LvermA Sachsen)
licenseCode	Lizenzcode des Servers. Wird von Fa. M.O.S.S. je Installation generiert.
licenseFeat	Liste der lizensierten Features, die der Server in dieser Installation unterstützt. Die Liste ist eine Folge von Nummern, deren Bezeichnung bzw. Bedeutung in der Datenbank hinterlegt ist.
serverID	Identifikation des Servers. Mit dieser ID werden die Serverdaten aus der Datenbank gelesen. Diese ID stellt sicher, daß mehrere Server auf die gleiche Datenbank zugreifen können (aber nicht alle Tabellen sind gleichzeitig verwendbar!)
serverGeo	Enthält den Identifikationstext, der in der DB unter der serverID für diesen Server definiert wurde.
docHanderFailMode	Enthält Schlüsselworte, die anzeigen, wie der GeoServer reagieren soll, wenn die Installation Dokumente per RelayServer und `dokFileHandler` unterstützt: ***terminateServer*** Wenn kein Zugriff auf den RelayServer möglich (d.h. es kann keine Verbindung aufgenommen werden), dann wird Startvorgang mit einem Fatalfehler abgebrochen. **keepAlive** Wenn kein Zugriff auf den RelayServer möglich, dann nur eine Warnung ins Protokoll schreiben, aber wieter hochfahren. Es können dann Dokumente dieser Typen nur indirekt über die vorhandenen Dokumente im

	Cache genutzt werden.
infoMessage	Allgemeine Meldung, die Clients abfragen können, z.B. für "`Message of the Day`"
loginEnabled	Flag ob (weitere) Login's zugelassen sind oder nicht. Wird dieser Eintrag auf 0 gesetzt, werden eingehende Login-Versuche abgewiesen, d.h. es wird eine Meldung geschickt und darauf gewartet, daß der Client seine Verbindung abbaut, sonst wird die Verbindung vom Server abgebaut
loginRestrictions	Wenn vorhanden, dann dient dies zur Überprüfung von speziellen Login-Einschränkungen. Diese Komponente enthält eine Liste der serverspezifischen Logineinschränkungen. Derzeit sind die folgenden Tests möglich: **sameUser:n** der gleicher Benutzer darf sich nur *n*-mal einloggen **acl:acl** allen Benutzern nur die hier angegebenen Rechte 'verpassen', dadurch kann ein Server zB. in reinen Auskunftsmodus geschaltet werden (mobilServer)
pendingShut	Flag zeigt an, daß der Server geruntergefahren werden soll. Es werden die Clients dann beim nächsten Zugriff/Request entsprechend informiert.
genNr	Generationsnummer für DokumentID-Generierung (wird benutzt um gleiche GeoCodes zu unterscheiden)
seqNr	Sequenznummer für DokumentID-Generierung (wird benutzt um gleiche GeoCodes zu unterscheiden)
mapSeqNr	Lfd. Sequenznummer für unterschiedliche Übersichtskarten (zur Speicherung der Informationen)
serverPort	Portnummer, die der Server für den Aufbau der Verbindungen von Clients benutzen soll. Alle Clients müssen mit dieser Portnummer eine Verbindung zum Server aufbauen (Dazu muß der Client noch den Hostnamen des GeoServers wissen). Wenn der GeoServer diesen Port belegt, wird intern die Socketverbindung im Zustand/Komponente 'serverSock' hinterlegt.
xfrReadBlockSize	Blockgröße, die beim Transfer von Dateien vom Server zum Client benutzt werden soll.
xfrWriteBlockSize	Blockgröße, die beim Transfer von Dateien vom Client zum Server benutzt werden soll.
aliveTime	Bei Start des Servers wird hier die Startzeit in Sekunden eingetragen, so daß man feststellen

	kann, wie lange der Server bereits aktiv ist.
logLevel	Protokollierungslevelnummer. Alle Meldungen, die größer diesem Protokollierungslevel sind, werden ausgegeben. Die Protokollierungslevel sind im Array logLevel hinterlegt, so daß symbolische Namen wie *panic, alert, critical, error, info, debug* usw. benutzt werden können.
debugTraceback	Flag, der anzeigt, ob bei einem Programmfehler, der per TcL-Befehl "`catch`" abgefangen wurde, trotzdem ein Traceback ausgegeben werden soll. Dies ist zur Fehlersuche hilfreich.
logFID	Ausgabedatei-Datei ID für Protokollierungsausgaben (default: `stdout`)
overviewFile	vollständiger Pfadname einer zu benutzenden Defaultübersichtskarte. Dieser Eintrag ist optional. Wenn vorhanden, dann muß auch die Komponente "*overviewFile,worldBox*" vorhanden sein. Damit kann eine spezielle Übersichtskarte als Startkarte definiert werden, denn sonst würde der GeoServer eine Übersichtskarte mit gleichem Namen wie die SGD-Datei, jedoch mit Endung ".tif" benutzen.
sgdFilename	vollständiger Pfadname zur Geodatenbasis. Wird die Datei geöffnet, dann werden die internen Informationen *sgdHandle*, *sgdWorldBox* und *sgdMeridian* mit den Werten aus der Datenbasis initialisiert. Man beachte, daß die GeoBase per Recovery wiederhergestellt werden kann, deshalb sollten hier keine "privaten" Ladeaktionen stattfinden, die nicht mitprotokolliert werden und damit bei einem Recoveryvorgang verloren gehen.
sgdMasterFile	vollständiger Pfadname zur Geodatenbasis, mit der der GeoServer installiert wurde. Diese SGD-Datei wird als "*safe*" für den Recovery-Vorgang angesehen. Alle Einträge, die bereits in dieser Datei enthalten sind und nicht in den Crossreferenztabellen der DB hinterlegt sind, bleiben beim Recovery erhalten.
sgdDoBackup	Flag ob beim Start des Servers ein Backup der SGD-Datei erstellt werden soll. Ist diese Komponente ungleich 0, erfolgt ein Sicherung.
sgdPntDelta	Angabe des Bereichs, der um eine Punktanfrage

	aufgespannt werden soll (alle Dokumente im Radius sgdPntDelta um den angegeben Punkt werden gesucht). Angabe in Weltkoordinaten.
sgdMercator	Flag, der anzeigt, ob die SGD-Datei die Daten im GK-System hinterlegt bzw. die Passpunkte im GK-System vorliegen.
mcWorldBox	Weltkoordinatenbox in der Form "xlu ylu xro yro". Diese Box limitiert den Koordinatenarbeitsbereich der GeoServer-Instanz. Alle Georeferenzen müssen innerhalb dieser Box liegen.
allowSGDdown	Flag, wenn ungleich 0, dann erfolgt ein Schließen der SGD-Datei nachdem alle Clients sich vom Server abgemeldet haben.
countSGDdown	Anzahl der reconnects auf die SGD-Datei (dieser Zähler wird nur inkrementiert, wenn der Flag `allowSGDdown` != 0)
attrTable	Name der Attributtabelle in der Geodatenbasis, über die die Dokumenteinträge gefunden werden können. Nur Grafiken, die Einträge in dieser Tabelle besitzen, können georeferenziert gefunden werden.
attr2DokTyp	Attributfeldname in der Attributtabelle der Geodatenbasis zur Bestimmung des Dokumenttyps. Der Dokumenttyp wird in dieser Spalte der Tabelle hinterlegt.
attr2DokID	Attributfeldname in der Attributtabelle der Geodatenbasis zur Bestimmung des Dokumentennamens. Hier wird die DokumentenID hinterlegt, sie muß eindeutig innerhalb der SGD-Datei sein.
dbConnect	Beschreibung der Datenbank und der Art des Zugriffs auf die Datenbank. Der Eintrag stellt eine Liste von Informationen dar, wobei der 1. Begriff die Datenbankzugriffsart beschreibt (erlaubt sind z.Zt.): • **ODBC** Zugriff per ODBC-Package, es muß der Name der Datenquelle (DSN) als 2. Angabe vorhanden sein. Nur unter WinNT • **DBSRV** Zugriff per DB-Server über Socketkommunikation. Es muß als 2. Angabe der Hostname des DB-Servers (oder leer falls auf dem gleichen Host wie der GeoServer) und die 3. Angabe die Portnummer des DB-Servers angegeben werden. • **IFXDB** Zugriff per Informix-Package Isqltcl. Es muß

	der Name der Datenbank (dbspace) angegeben werden. Der Informix-Server wird aus der Environmentvariablen INFORMIX-SERVER geholt, falls der Name der Datenbank nicht den DB-Server mit enthält (z.B. rissdb@odsappl_tcp, dann wird der DB-Server odsappl_tcp herangezogen und dort der dbspace rissdb aktiviert) • **ORADB** Zugriff per Oracle-Package oratcl. Es muß der Name der Datenbankverbindung angegeben werden. Als 3. Parameter kann eine Liste mit Benutzername, Password und DB-Server angegeben werden. Wurde eine Verbindung zur DB aufgebaut, werden die internen Zustandsinformationen dbHandle, dbID und dbStream usw. gesetzt.
dbKeepAlive	Flag, der bei einem Zugriff auf einem DB-Server anzeigt, ob bei Beendigung des GeoServers (Shutdown) auch der DB-Server beendet werden soll (ist dbKeepAlive = 0, dann wird ein Shutdown-Request an den DB-Server bei Shutdown des Geo-Servers gesendet).
cacheTimeout	Angabe in Sekunden, nach Ablauf dieser Zeitspanne werden interne Cache-Daten neu von der DB eingelesen. Jede Information im Cache erhält einen Zeitstempel, so daß für jede Information individuell das "Alter" festgestellt und entsprechend der Cache aktualisiert wird
allowAllUsers	Flag, wenn ungleich 0, dann erfolgt keine Zugriffsprüfung mittels der Userinformation in der Datenbank (s. allowAllUsrAcl)
allowAllHosts	Flag, wenn ungleich 0, dann erfolgt keine Zugriffsprüfung des Clienthosts anhand der Datenbankinformationen.
allowAllUsrAcl	Zugriffsrechte, falls allowAllUsers=1, die jedem Client dann zugeordnet werden. Ist dieses Element nicht definiert, wird "Auskunft"-Berechtigung gestattet.
allowAllHostAcl	ID der Zugriffsrechte, falls allowAllHosts=1, diese AclID wird jedem Client zugeordnet. Ist dieses Element nicht definiert, dann bleibt die Zugriffsberechtigung gemäß Usereintrag (bzw. `allowAllUsers` mit `allowAllUsrAcl`) erhalten.

accFailLimit	Maximale Anzahl von Versuchen, die ein Client machen darf um priviligierte Befehle oder Logins auszuführen, bevor die Verbindung abgebrochen wird.
applLoginMode	Defaultwert, der beim Einrichten eines neuen Benutzers (addSimple) für den Loginvorgang in der DB definiert wird. Wenn 1, dann muß der Applikationslogin durchlaufen werden, sonst nicht.
applLoginCmd	Information darüber, welche Login-Applikation zu aktivieren ist. Die Befehle müssen alle mit der Applikationsschnittstelle (applLoginCmd, applLoginDone, applLoginBegin) in COMX realisiert werden.
applCountXFR	Informationen darüber ob ein spezielle Zähl-programm/-verfahren zu durchlaufen ist, wenn ein Dokument transferiert wurde. Dies stellt den Namen einer Prozedur dar, die mit der Methode "recordDokID" aktivier wird.
fileStorage	Flag ob Dokumentdateien in der Datenbank zu speichern sind (kann später auch für jede Do kumentart einzeln definiert werden)
fileLocal	Flag ob Dokumentdateien im lokal Filesystem (Plattenzugriff bzw. Netwerkzugriff) erreicht werden könenn, wenn ja kann der Dateitransfer per Betriebssystem durchgeführt werden.
filePath	Suchliste, wo die Dokumentdateien im Serverfilesystem zu finden sind. Nur nötig, wenn Daten im Filesystem hinterlegt werden.
fileCacheMax	Maximale Anzahl der Dateien, die im FileCache vorgehalten werden
dokidGenerator	Angabe einer EASI/Tcl-Prozedur, die aktiviert wird, um aus den DB-Angaben eine DokID zu erzeugen. Diese ID muß eindeutig für alle Dokumente sein (z.B. Generierung einer Belegkennung aus den Sachdaten für ein Dokument). Beispiel für einen Generator ist die Prozedur "dasy_marDokID"
zipDokID	Sollte die DokID, die für ein Dokument abgeleitet wird, nicht in die SGD-Attributtabelle passen, dann muß die DokID komprimiert werden, mit dieser Komponente kann die EASI/Tcl-Prozedur definiert werden, die eine DokID so komprimiert, daß sie als SGD-Attribut benutzt werden kann, aber trotzdem eindeutig ist und auch dabei keine Daten verloren gehen. Beispiel dasy_marZipDokID
unzipDokID	Sollte die DokID, die in die SGD-Datei als Attribut hinterlegt wird vorher komprimiert worden sein,

	dann muß diese DokID bei einer Selektion wieder dekomprimiert werden, dies geschieht dann mit der hier anzugebenden Prozedur. Beispiel `dasy_marUnzipDokID`.
file2ppGenerator	Es werden bei TIFF-Dateien, die auch als Übersichtskarte benutzt werden können, die Transformationsdaten aus der TIFF-Datei gelesen (TIFF-Datei muß die GeoInformation entweder gemäß MOSS-Methode, GeoTIFF oder als AED-Geotags enthalten). Sind jedoch in der TIFF-Datei keine GeoInformationen abgelegt, dann kann man erreichen, daß mit Hilfe einer EASI/Tcl-Prozedur die GeoInformation aus dem Datei- oder Dokumentnamen abgeleitet wird. Die dazu zu benutzende Prozedur ist in dieser Komponente anzugeben (z.B. kann aus dem Dokumentennamen von Rahmenkarten, die die Blattnummer beinhalten die GeoInformation generiert werden, z.B. dasy_marTif2PP). Der Generator wird aktiviert per `geoState(file2ppGenerator) <file> ?boxFlag?` Ist BoxFlag vorhanden und ungleich Null, dann wird nicht wie sonst die linke untere und rechte obere Ecke als Koordinaten geliefert, sondern es werden alle 4 Eckkordinaten geliefert. Sinnvoll bei einer Georeferenzierung um in einem Paßpunktdialog den Weltkoordinaten die Bildkoordinaten zuzuordnen.
tiffppinfo	Es werden die Geoinformationen mit Hilfes dieses Programms aus den Bild-Dateien (TIFF) gelesen. Es können hier auch spezielle Optionen für das Programm mit angegeben werden. Beim Auslesen der Information wird nur noch der Dateiname angehängt.
clientNum	Interne Zustandsinformation, es wird bei jeder Clientverbindung diese Zahl erhöht. Alle Daten des zuletzt angemeldeten Clients werden im Zustandsarray "geoClients" unter dieser Nummer geführt.
curClients	Interne Zustandsinformation über die Anzahl der aktuell aktiven Clients (Anzahl der offenen Verbindungen zu Clients). Diese Information wird benutzt um die maximale Anzahl der Clients zu begrenzen.
inActiveTimeOut	Zeit in Sekunden, die ein Client inaktiv sein kann, ohne daß bei der Prüfung auf onaktive Clients, dieser Client automatisch ausgeloggt wird.

	Der Inaktivitätstest wird nur beim Login eines neuen Clients durchgeführt oder per ADMN inative check initiiert
maxClients	maximal zulässige Anzahl von Clients, die gleichzeitig mit dem Server verbunden sein dürfen. Wird per Lizenzcode festgelegt.
typMapStamp	letzter Zeitstempel für die Liste der Dokumenttypen, wird intern benutzt um den Typlisten-Cache mit den DB-Typen abzugleichen.
tblDoAccList, **tblDoAccList,** **tblDoErrMsgT,** **tblDoGeoHotSpT,** **tblDoGeoXPntsT,** **tbl,DoGeoXTableT,** **tblDoGeschJahrT,** **tblDoMsgMapT,** **tblDoProtokT,** **tblDoSysDataT,** **tblDoTrustedHostsT,** **tblDoTypDefT,** **tblDoUsersT,** **tblWegaColumns,** **tblWegaEnums,** **tblWegaProcs** **tblWegaTypes,** **tblWegaLocks**	diese Komponenten werden dazu verwendet um die GeoServer-Tabellennamen auf die realen Tabellennamen in der DB umzusetzen. Mit Hilfe dieser Umsetzungsmethode ist es möglich, daß mehrere GeoServer auf der gleichen DB operieren können.
languageID	Kennung der Sprache, in der ein Client seine Meldungen vom Server erhalten wird. Diese Spracheinstellung kann vom Client geändert werden (siehe geoClient-Information languageID).
maxQryDokCount	Maximale Anzahl von Treffen bei einer Sachdatenabfrage, dieser Wert wird als Defaultwert an jeden Client vererbt. Der Client selbst kann (sofern er die Rechte dazu hat) dieses Limit verändern.
sizeTolerance	Angabe einer Prozentzahl (0..100), die benutzt wird, um die DIN-Größe eines Bildes zu bestimmen. Ein Bild darf um diese Prozentangabe größer sein um trotzdem noch dem nächstkleineren DIN-Format zuzuordnen.
clientScriptsPath	Suchliste mit Pfadangaben, wo Clientskripte, z.B. VB-Skripte, hinterlegt sind. Diese Skripte werden vom Client abgefragt und falls die Aktualität nicht gegeben ist, vom Server runtergeladen. Die Suchliste wird von links nach rechts abgearbeitet und die erste gefundene Datei wird dem Client übermittelt.

clientModulesPath	Suchliste mit Pfadangaben, wo Clientmodule, zur Aktualisierung des GDM Clients hinterlegt sind (z.B. DLL's). Diese Module werden vom Client abgefragt und falls die Aktualität nicht gegeben ist, vom Server runtergeladen. Die Suchliste wird von links nach rechts abgearbeitet und die erste gefundene Datei wird dem Client übermittelt. Der Abgleich der Module des Clients erfolgt anhand einer INI-Datei "`GDM-Client.ini`". Hier sind alle Modulinformationen und Versionsnummer enthalten, so daß der GDM-Client seine eigenen Module gegenüber den zur Verfügung stehenden aktualisierten Modulen vergleichen kann.
vbsTransferMode	Transfermodus, der beim Transfer von Skriptdateien als default benutzt wird. Ist dieser Eintrag nicht vorhanden, dann wird "forceServer" verwendet. Siehe Befehl QVBS.
vbsDevList	Liste von Patterns. Es wird bei dem Befehl QVBS geprüft ob in Clientinformationen die Variable geoClient(*,vbsDevList) vorhanden ist, wenn ja wird dann diese Patternliste benutzt um Dateien, auf die Pattern zutreffen, im KeepLocal-Modus zu melden. Ist dieser Eintrag nicht vorhanden, dann wird geprüft ob ein Eintrag in geoState(vbsDevList) vorhanden ist, wenn ja wird diese Liste benutzt.

geoClient-Informationen

Jeder Client, der mit dem Server verbunden ist, erhält eine ClientID, diese ID wird als Index im Array *`geoClient`* benutzt, um dort Client-spezifische Informationen zu hinterlegen. Fast alle dieser Informationen haben internen Charakter und sollten nicht vom Client bzw. von anderen Clients direkt verändert werden können. Die ClientID bleibt bis zum Neustart des GeoServers eindeutig (wird für jede Clientverbindung inkrementiert)

Komponente	Bedeutung
uid	Useridentifikationsnummer des Clients.
usr	Benutzerkennung des Client, wie beim Login angegeben.
infoMessage	Allgemeine Meldung, die dem eingeloggten Benutzer geliefert wird, sofern der Client dies abfragt. Nach einer Abfrage wird dieser Eintrag wieder gelöscht.

languageID	Kennung der Sprache, in der der Client seine Meldungen vom Server erhalten möchte, Z.Zt. sind die Kennungen: • ger – deutsch • eng – englisch möglich. Dieser Eintrag kann während der Clientsitzung auf eine andere Sprache umgestellt werden
accumZeit	akkumulierte Login-Zeit des Clients. Diese Angabe wird nur von Zeit zu Zeit akutalisiert
aclInfo	Access Control List Information, beschreibt die Benutzerrechte des Clients.
accFail	Anzahl der Versuche einen priviligierten Befehl ohne Berechtigung auszuführen. Diese Information wird dazu verwendet, um nach einem bestimmten Limit, die Clientverbindung abzubrechen (siehe geoState *`accFailLimit`***)**
applLogin	Information darüber, welche Login-Applikation zu durchlaufen ist, bevor der Client überhaupt "akzeptiert" wird. Ist dieser Wert "0" dann ist keine Login-Anwendung definiert. Hier wird u.a. das "Geschäftsbuch" eingehängt. Der Client wird dann in einen neuen Verbindungszustand gesetzt, der solange Gültigkeit hat, bis die Login-Applikation erfolgreich durchlaufen wurde.
queryCnt	Anzahl der Anfragen, die der Client durchgeführt hat.
rdfilCnt	Anzahl der erfolgreich gelesenen Dokumente in dieser Sitzung
wrfilCnt	Anzahl der vom Client erfolgreich eingebrachten (eingerichteten) Dokumente in dieser Sitzung
unfilCnt	Anzahl der fehlerhaft durchgeführten Dokumentendateitransfers
countThis	Flag, der anzeigt ob ein Transfer in die Accountinginformation zu übernehmen ist. Wenn ja, sind weitere Informationen angehängt: "`countFlag dokTypID dokID dinSize isGeoref`"
ctime	Zeitstempel des Clients, zu diesem Zeitpunkt wurde die Verbindung (Login) aufgebaut (connect time).
atime	Akkumulierte Zeit, die der Client bereits mit dem Server verbunden ist (diese Information wird nur zum Zeitpunkt des Logouts erzeugt.

expire	Zeitpunkt, wann das Login-Password seine Gültigkeit verliert.
state	Aktueller zustand der Client-Verbindung, je nach Zustand sind nur bestimmte Befehle erlaubt.
collHandle	Falls eine Query gerade aktiv ist, dann ist der Handlename der Collection hier angegeben.
collState	Lesezustand einer geöffneten Collection.
sqlHandle	Falls SQL-Query gerade aktiv, dann ist hier der SQL-Queryhandle (Cursor) hinterlegt.
sqlState	Falls SQL-Query in Progress, dann wird hier der Zustand der Query eingetragen (begin, extend, execute, cursor)
readState	Informationen, falls Client gerade dabei ist, eine Datei vom Server zu lesen.
readID	fileStream, der für den Lesevorgang benutzt wird, ist nur während des Zustands „readState aktiv“ gültig.
writeState	Informationen, falls Client gerade dabei ist, eine Datei an den Server zu senden.
writeID	fileStream, der für den Schreibvorgang benutzt wird, ist nur während des Zustands „writeState aktiv“ gültig.
sock	Socket-FileId, die zur Kommunikation mit dem Client zu benutzen ist.
host	Hostname wie Client bei Verbindungsaufbau gesendet (server connect-Information)
port	Portnummer des Clients wie bei Verbindungsaufbau gespeichert.
incmd	Die vom Client gesendete Message, die weiter analysiert wird, um so den Befehl auszuführen
cmd	Nach dem Parsevorgang wird hier der auszuführende Befehl gespeichert.
data	Nach dem Parsevorgang werden hier die Daten, die mit dem Befehl geschickt wurden, hinterlegt
overviewFile	aktuelle zu benutzende Übersichtskarte für diesen Client. Ist diese Komponente nicht belegt wird die Servereinstellung benutzt. Wenn vorhanden, dann muß auch die Komponente "overviewFile,worldBox" vorhanden sein.
maxQryDokCount	Maximale Anzahl von Treffern bei einer Sachdatenabfrage. Ein Ergebnis mit mehr Treffern wird abgewiesen.

vbsDevList	Siehe vbsDevList in der geoState-Beschreibung. Es wird ein Security-Alert ausgelöst, wenn dieser Eintrag vorhanden ist.

Der GeoServer speichert noch weitere interne Informationen, die jedoch nur für interne Zwecke zu benutzen sind und daher nicht hier aufgeführt werden, z.B. interne Caches.

Datums- und Zeitangaben im GeoServer

Es wird ein eigenes Datums- bzw. Zeitformat verwendet, daß auch bei Zeitangaben in Datenbankspalten verwendet wird. Um solche Zeitangaben erzeugen zu können benötigt man das Tcl-Package `EcmDoRIS`.
Das Package definiert beim Laden den Befehl "***ecmdoris***". Dieser Befehl besitzt wiederum viele Subbeefehle, u.a. den Subbefehl "*datetime*" zur Umrechnung und Formatierung von Zeitangaben. Um "schnell" eine Zeitangabe zu erhalten, geht man wie folgt vor:

- Starten einer Tcl-Umgebung (**`tclsh`** oder **`wish`**)
- Laden des Packages
  ```
  package require EcmDoRIS
  ```
- aktuelle Zeit nach GDM-Zeitangabe:
  ```
  ecmdoris datetime now
  ```
- GDM-Zeitangabe invertieren:
  ```
  ecmdoris datetime -invers <value>
  ```

Definition von Dokumenttypen im GeoServer

Es Dokumente verschiedenen Typs im System verwaltet werden. Die Dokumente werden nach Dokumenttyp unterschieden. Jedem Dokumententyp können verschiedene Eigenschaften zugeordnet werden

- Dokumententypbeschreibung
- Dateityp bzw. Datenformat (z.B. TIFF, JPEG, Word-Dokument)
- Zusätzliche alternative Datei- bzw. Datentypen, die der Dokumententyp ebenfalls erlaubt
- Foliendefinition falls der Dokumententyp georeferenziert ist
- Zeitpunkt wann zuletzt, die Typdefinition geändert wurde
- Ob der Dokumententyp aktiv ist (man kann auch Typen definieren, die erst später aktiv gechaltet werden, d.h. dem WEGA-GDM Client gegenüber sichtbar werden)
- Ob das Dokument georeferenziert ist (die Georeferenz kann ein Punkt, Rechteck oder Polygon sein. Ist das Feature „LassoObjects" aktiviert, dann kann die Referenzierung aus mehreren Polygonzügen (Lassoschlingen) bestehen. Die Gesamtzahl der Punkte darf 32767 Punkte nicht überschreiten)

- Ob dieses Dokument als Karte verwendet wird, d.h. dieses Dokument kann als Hintergrund für eine Geoselektion verwendet werden (z.B. ALK-Ausschnitt, TK10-Karte, Rahmenkarte usw.).
- Wenn das Dokument als Karte verwendet wird, dann kann festgelegt werden, wie Geoinformationen des Dokumentes erhalten werden können bzw. wenn eine Karte eingerichtet wird, wie dann die zugehörige Georeferenz erhalten werden kann (z.B. GeoTIFF Feature „GeoTIFF“ muß aktiviert sein, Headerdatei, Passpunktedatei oder Ableitung der Informationen aus dem Dokumentennamen)
- Ob diesem Dokumententyp eine Datei zugeordnet ist. Ist dies nicht der Fall, müssen die Informationen zu dem Dokument irgendwie berechnet werden (computed document, wird derzeit nicht vollständig unterstützt)
- Ob das Dokument schreibgeschützt ist (Urkunden-Charackter). Es werden Änderungs- oder Löschversuche abgewiesen.
- Ob das Dokument zu einer Gruppe von Dokumenten gehört (Dokumentenmappe). Nur möglich, wenn das Feature „ConnectedObjects” aktiviert ist. Es werden dann bei der Auswahl eines Dokumentes auch alle anderen zusammengehörenden Dokumente bei einer Abfrage verfügbar (z.B. verschiedene Unterlagen zu einem Baugesuch könnten als Dokumentenmappe zusammengefasst werden).
- Informationen zu Sachdaten, die ein Dokument besitzt und anhand derer das Dokument auch gesucht werden kann
- Die Art der Speicherung des Dokumentes (im Dateisystem, in Datenbank als Blob oder per externem Zugriffsverfahren)
- Informationen darüber, wie der Zugriff auf das Dokument zu protokollieren ist (z.B. Geschäftsbuch, Abrechnungsmodi)
- Informationen, die der GeoClient zur Anzeige, zum Ausdrucken, zum Auflisten der Sachdaten, bzw. für Applikationen benötigt.
- Ob der Dokumententyp weitere Dateien beinhaltet kann, die als sogenannte Attachments bezeichnet werden. Attachments sind im Prinzip Zip-Archive in denen zusätzliche Dateien gespeichert werden (man kann jedoch nicht nach speziellen Dateien im Archiv suchen, sondern nur abfragen, ob ein Dokument ein Attachment besitzt oder nicht. Attachment sind nur dann möglich, wenn das feature „Attachments“ im GeoServer aktiviert ist.
- Ob ein Dokumententyp auch Annotationen erlaubt, nur Rasterbilder können Annotationen erhalten (manchmal auch als Redline-Funktion bezeichnet). Annotationen sind zusätzliche Bildebenen, die über das Bild eingeblendet bzw. gezeichnet werden können, um z.B. geänderte Situationen zu kennzeichnen. Annotationen sind nur möglich, wenn das Feature „Annotation“ im GeoServer aktiviert ist.

Alle Dokumente werden am GeoServer in der Datenbank definiert (Tabelle ***doTypDefT***). Es können pro GeoServerinstanz maximal 32767 Dokumenttypden definiert werden. Der GeoClient holt sich alle Informationen über die vorhandenen Dokumenttypen und deren Eigenschaften beim login-Vorgang.

Die Dokumentdefinitionen sind in der Datenbank hinterlegt, wobei der Zugriff auf Sachdaten zu dem Dokument über ein zentrales Dictionary gesteuert wird. Damit die Zugriffe auf die Dokumentinformationen möglichst schnell ablaufen, werden

diese Informationen im Speicher geladen und in bestimmten Zeitintervallen mit der Datenbank wieder abgeglichen.

Einbringen eines neuen Dokumententyps ohne Georeferenz

Dokumententyp ohne Georeferenz, d.h. es wird das Dokument nur mit Sachdaten gespeichert, eine zusätzliche Georeferenz ist nicht definiert. Hier im Beispiel werden die Seiten des Flurbuchs gescannt und eingebracht.

Man sollte zuerst unter dem Verzeichnis "<wegainstallPath>/server/sql" ein Unterverzeichnis der Applikation bzw. des Kunden einrichten, z.b. .../sql/mh und dort alle kunden- bzw. applikationsspezifische Dateien speichern.

Bei Erzeugen der Datenbanktabellen wird dann zuerst der allgemeine Teil definiert und gefüllt und danach wird in das kunden- bzw. applikationsspezifische Verzeichnis gewechselt und alle dort hinterlegten Tabellendefinition bzw. Daten geladen.

- zuerst Definition der DB-Tabelle für die Sachdaten des Dokumentes. Diese Definition wird in einer Tcl-Datei hinterlegt, deren Namen den Namen der DB-Tabelle wiedergibt (es darf nicht mit "values" enden), z.B. mhflurbucht.tcl:

```
sqlCommand {CREATE TABLE mhFlurbuch (
   DokID          CHAR(28),
   GemNr          SMALLINT,
   FlurBez        FLOAT,
   SeitenNr       SMALLINT,
   LadeDatum      INTEGER
)}
```

- nun für die typischen Suchanfragen auf den Elementen jeweils einen Index in der Definitionsdatei definieren:

```
sqlCommand {CREATE UNIQUE INDEX mhFlurbuchUX1 ON mhFlurbuch (DokID)}
sqlCommand {CREATE      INDEX mhFlurbuchIX2 ON mhFlurbuch (GemNr)}
sqlCommand {CREATE      INDEX mhFlurbuchIX3 ON mhFlurbuch (FlurBez)}
```

- Die neue Tabelle wird dem GDM bekanntgemacht, indem die Tabelle in der Crossreference-Tabelle ***'tblWegaGDM'*** eingetragen wird. Hier wird eine eindeutige Tabellenidentifikation (tabID) geliefert bzw. benutzt, diese neuen Werte sollten in der Datei "tblwegagdmvalues.tcl" hinterlegt werden:

```
catch {unset values}
set values {
    {106, 'mhFlurbuch', 'Flurbuch mit Übersicht der Flurstücke', \
     '1', 36451}
}

foreach row $values {
    sqlCommand "INSERT INTO tblWegaGDM VALUES ($row)"
}
```

- Nun können die Spaltendefinitionen der Tabelle in GDM registriert werden. Hierzu werden die Typangaben aus der Tabelle **'tblWegaTypes'**

herangezogen (z.B. CHAR wird zu GDMSTRING und damit zu 1). Jede Spalte, die hier registriert wird, muß mit dem DB-Spaltennamen, wie beim Erzeugen der Tabelle definiert, identifiziert werden. Aus der eindeutigen Spaltenidentifikation kann dann umgekehrt wieder auf die Zieltabelle (per tabID) zugegriffen werden. Hier werden die für die Spalten die Texte, definiert, die der GDM-Client als Spaltennamen bei der Ausgabe in Tabellen/Formularen etc. benutzt, während für die interne Kommunikation der reale Spaltenname (bzw. ColID) benutzt wird (die Einträge sind in der Datei `tblWegaSyColumnsvalues.tcl` zu hinterlegen):

```
catch {unset values}
set values {
   {40, 106, 'DokID', 1, 0, 'Flurbuch DokID','Dok-Kennung Flurbuch', 0}
   {41, 106, 'GemNr', 2, 0, 'GemarkungsNr.', 'Gemarkungsnummer', 0}
   {42, 106, 'FlurBez', 11, 0, 'Flurbezeichnung', 'Flurbezeichnung', 0}
   {43, 106, 'SeitenNr', 2, 0, 'SeitenNr', 'fortlaufende SeitenNr', 0}
   {44, 106, 'LadeDatum', 8, 0, 'LadeDatum', 'LadeDatum', 0}
}

foreach row $values {
    sqlCommand "INSERT INTO tblWegaSyColumns VALUES ($row)"
}
```

- Nun liegt alles in DB fest, wir können nun den neuen Dokumententyp in der Dokumenttyptabelle nachtragen. Hier wird eine eindeutige DokTyp-Identifikation und ein eindeutige DokTyp-Name vergeben (bzw. generiert). In dieser Tabelle wird festgelegt ob der Dokumenttyp
  - z.Zt. aktiv (d.h. enabled) ist, nur dann kann der GDM-Client diesen Dokumententyp verwenden.
  - georeferenziert ist.
  - eine Karte darstellt, d.h. für Koordinatenangaben bzw. -abfragen benutzt werden kann. Ein solcher Typ kann dazu benutzt werden, um weitere Trefferabfragen in dieser Karte zu erhalten.
  - eine Datei ist. Ist dies nicht der Fall, dann wird von einem "computed-Dokument" gesprochen, d.h. die Informatioen (Sachdaten und/oder Geodaten) werden zur Lauf- bzw. Anfragezeit per Programm/SQL-Anweisung etc. ermittelt und an den GDM-Client geliefert.
  - ob diese Dokumente gelöscht werden dürfen.
  - als ein Container für andere Dokumente dient.
  - wie das Dokument gespeichert ist (im Filesystem erreichbar, per RelayServer erreichbar oder als BLOB in der DB gespeichert usw.)
  - welche Sachdatentabelle verwendet wird. Es kann anstelle der Tabellenidentifikation (d.h. ein positiver Wert, der die Sachdatentabelle per ID in der Tabelle *tblWegaGDM* definiert) der Wert 0 oder ein negativer Wert angegeben werden:
    - Der Wert 0 signalisiert, daß keine Sachdaten in der DB für diesen Dokumententyp vorhanden sind.
    - Ein negativer Wert definiert einen Skript-gesteuerten Zugriff auf die Sachdaten. Die ID muß dann in der Tabelle der GDM-Skripte

(`tblWegaProcs`) definiert sein (s. "Sachdatenzugriff per DokTyp-Prozedur")

diese Definition wird in der Datei "`dotypdeftvalues.tcl`" hinterlegt:

```
catch {unset values}
set values {
   {21, 'Flurbuch', 'tif', 'Flurbuch', 'Folie_18', 36451.50, \
     '1', '0', '0', '1', '0', '0', 106 }
}

foreach row $values {
    sqlCommand "INSERT INTO DoTypDefT VALUES ($row)"
}
```

Einbringen eines neuen Dokumententyps mit Georeferenz

Dokumententyp mit Georeferenz, d.h. es kann das Dokument per Selektion in einer Übersichtskarte selektiert werden. Alle Dokumente müssen im Koordinatenraum des GeoServers liegen. Dieser Koordinatenraum wird bei der Definition des GeoServers festgelegt. Alle Dokumente, deren Georeferenz außerhalb dieses Koordinatenraums liegen, werden abgewiesen.
Sind die Koordinaten in verschiedenen 'Koordinatensystemen', z.B. Dokumente in verschiedenen Gauß-Krüger-Streifen, dann muß eine entsprechende Umrechnungsfunktionalität im Server zur Verfügung stehen, damit diese Koordinaten in ein "Super-Koordinatensystem" (gemeinsames Koordinatensystem) umgerechnet werden können.

Sind für den Dokumententyp Sachdaten notwendig, so sind diese Angaben in der DB entsprechend dem Punkt "Einbringen eines neuen Dokumententyps ohne Georeferenz" einzubringen.

- Festlegung der Umsetzung von Sachdaten in eine eindeutige DokID, sofern dies notwendig ist, z.B.:

  ```
  set geoSetup(dokidGenerator)   dasy_marDokID
  ```

  Diese Prozedur wird immer dann gerufen, wenn für angegebene Sachdateninformationen ein Belegname (=DokumentenID) generiert werden soll. Ist eine solche Prozedur nicht definiert, dann wird die DokID vom GDM-Client benutzt.

  Aufruf:
  ```
  dasy_marDokID <doktyp> <dbdaten>
  ```

  Parameter:

`<dokTyp>`	Dokumententyp, je nach Typ wird die DokID generiert.
`<dbdaten>`	Sachdaten als Folge von Listen, jedes Element enthält Sachdatenname (wie in der DB definiert) und den Wert

- Es kann nunmehr noch festgelegt werden, ob für den Dokumenten eine spezielle Prozedur zur Umsetzung der DokID auf die GeoBase bzw. einer GeoBase-ID in eine DokID, zu benutzen ist (meisten zum kompirimieren und

kodieren von bestimmten Sachdaten in der DokID, damit diese eindeutig wird und in GeoBase-Attributtabelle "hineinpaßt"), z.B.:

```
set geoSetup(zipDokID)          dasy_marZipDokID
set geoSetup(unzipDokID)        dasy_marUnzipDokID
```

Aufruf:

```
dasy_marZipDokID <doktyp> <dokID> [<dbdaten>]
```

Parameter:

`<dokTyp>` Dokumententyp, je nach Typ wird die zugehörige DokID generiert.

`<dokID>` DokID wie vom GDM-Client oder vom DokID-Generator generiert.
Es kann auch ein Leerstring angegeben werden, falls keine DokID definiert ist, dann muß aber <`dbdaten`> belegt sein.

`<dbdaten>` Sachdaten die vom GDM-Client via Formular angefordert und an den
GeoServer geschickt wurden.

- bei manchen Dokumenten ist es möglich die Georeferenz aus dem Dokumentennamen abzuleiten, insbesondere bei Rahmenkarten, die einem bestimmten regelmäßigem Numerierungsschema folgen. Daher kann für den Dokumententyp eine weitere Prozedur definiert werden, die eine Umsetzung von dem Dokumentennamen auf die Georeferenz (z.B. die Eckkoordinaten einer Rahmenkarte) liefert. Diese Prozedur wird immer dann aktiviert, wenn der GeoServer die Georeferenzinformation des entsprechenden Dokumentes benötigt, z.B.:

  ```
  set geoSetup(file2ppGenerator) dasy_marTif2PP
  ```

  Aufruf:

  ```
  dasy_marTif2PP <tiffdatei>
  ```

  Parameter:

  `<tiffdatei>` Dokumentendateiname, es darf eine Pfadangabe vorhanden sein.
  Return-Wert:
  `<WorldBox>` umschreibendes Rechteck des Dokumentes in Weltkoordinaten in der Form "`Lux Luy Rox Roy`"

Alle diese Prozeduren sind beim Hochfahren des GeoServers bereitzustellen. Diese Angabe werden daher in der GeoServer-Parameterdatei hinterlegt.

- Nun liegt alles für die Georeferenzierung fest, wir können nun alle weiteren Daten in DB einbringen.

- Abschließend wird in der Datenbank der neue Dokumententyp in der Dokumenttyptabelle eingetragen. Hier wird eine eindeutige DokTyp-Identifikation und ein eindeutiger DokTyp-Name vergeben (bzw. generiert):

```
catch {unset values}
set values {
    {12, 'RissDatei', 'tif', 'Rissdokument MH', 'Folie_10', 36328.50, \
      '1', '1', '0', '1', '0', '0', -1 }
}

foreach row $values {
    sqlCommand "INSERT INTO DoTypDefT VALUES ($row)"
}
```

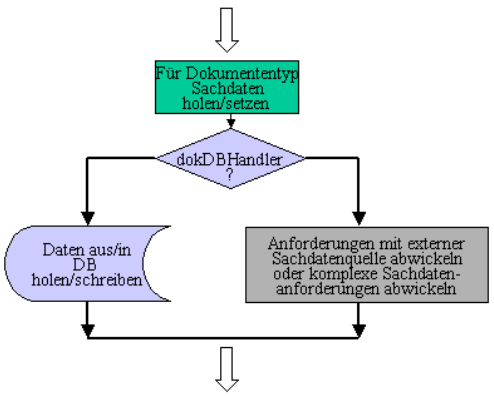
Sachdatenzugriff per DokTyp-Prozedur

Bei der Definition von Dokumenttypen ist es möglich, daß anstelle einer Datenbanktabelle, eine Prozedur benutzt werden soll, ein sogenanter `dbHandler`. Diese Prozedur muß dann zum Client hin eine DB-Tabelle "simulieren".

Der GeoServer liest aus der Dokumententypdefinition, welche Prozedur zu aktivieren ist (dies geschieht durch einen Indirektionsschritt, indem die Prozedur anhand einer Identifikation aus einer Prozedurtabelle geholt wird).

Überall dort, wo Sachdaten zu einem Dokument eingebracht bzw. geholt werden, ruft dann der GeoServer die angegebene Prozedur mit einem entsprechenden Methodennamen und den angehängten notwendigen Daten auf.

Die einzelnen Methoden müssen die angeforderten Daten bzw. die bereitgestellten Daten aufbereiten und in die DB speichern bzw. von der DB holen (als Beispiel diene die Implementation von `MHdokDB.tcl` zur Bereitstellung des Dokumententyps `'RissDatei'` für Mülheim a. d. Ruhr).



Manchmal möchte man Sachdaten zu einem Dokument holen, die nicht in der gleichen Datenbank enthalten sind, in der der GeoServer seine Daten hält, z.B. Zugriff auf das ALB, das auf einen Rechner und/oder in einer anderen DB läuft. Andererseits mögen die Sachdaten so komplex sein, daß die Standardfunktionalität des GeoServers nicht ausreicht.

Es wird daher immer wenn der Client bzw. eine Applikation Sachdaten zu einem Dokument benötigt bzw. Sachdaten einbringen will, wird bei jedem Dokumententyp geprüft, ob für diesen Typ ein sogenannter ***dokDBHandler*** definiert ist, wenn ja wird die Anfrage an diesen Handler weitergeleitet. Dieser Handler muß dann die Kommunikation mit der externen Sachdatenquelle abwickeln

45

Es werden die folgenden Methoden vom GeoServer benötigt:

columns

←`columns dokTyp`
→`{dokid GDMSTRING 0 {DokumentKennung}} {..}`

Auflisten der (virtuellen) Sachdatenspalten. Sind mehrere Tabellen beteiligt, sind die Spaltennamen mit einem entsprechenden Prefix zu versehen. Alle Sachdaten werden mit diesen (internen) Spaltennamen operieren bzw. diese benutzen. Pro Spalte wird folgende Information geliefert:

```
{myColName gdmDataType isList ColLabelString}
```

ColLabelString ist dabei der Spaltentext, der in Formularen zu benutzen ist. Der Flag 'isList' zeigt an, ob eine Liste von Werten erlaubt ist. Damit kann man an ein Dokument z.B. eine Liste von veränderten Flurstücken zuordnen. Die add/find/get-Methoden müssen dann diese Liste entsprechend in einzelne Sachdatensätze aufsplitten und in die Tabellen eintragen bzw. von dort auslesen.

count

←`count dokTyp`
→`dokCntTab1 dokCntTab2 ...`

Anzahl der Sachdatenrekords in den einzelnen Tabellen zurückliefern. Für jede beteiligte Tabelle ist ein *Count*-Wert zu bilden und alle zusammen als Liste zu liefern. Der *Count*-Wert der wichtigsten Tabelle sollte an 1. Stelle stehen.

countAll

←`countAll dokTyp`
→`dokCnt`

Anzahl der Sachdatenrekords in der Haupttabelle bestimmen und zurückliefern.

add

←`add dokTyp dokID dokData`
→`-empty-`

Sachdaten für ein neues Dokument in die DB eintragen. Es wird vorausgesetzt, daß bereits die Eindeutigkeit der dokID geprüft wurde. Die Sachdaten sind in Form einer geschachtelten Liste angegeben. Jedes Listenelement stellt einen Wert der Sachdatenspalte dar:

```
{myColName ColumnValue} ...
```

Ist die zugeordnete Spalte als Listentyp definiert, dann ist wird ein Listenelement mehrere *ColumnValues* aufweisen, z.B.

```
{myColName ColumnValue1 {} ColumnValue3 ...}
```

wobei "{}" als Platzhalter für eine leere Wertangabe zu benutzen ist. Die Anzahl der ColumnValues definiert wieviel Sachdatenrekords letztendlich in der DB abzuspeichern sind. Der n'te Rekord wird hierbei per

```
set recNr n'th
foreach colInfo dokData {
    set colValuelindex [lindex $colInfo $recNr}
```

```
}
```

ausgelesen.
Die Spaltennamen entsprechen hierbei den Spaltennamen, die mit der "columns"-Methode geliefert wurden. Die "add"-Methode muß diese Spaltennamen in die entsprechenden Tabellen und Tabellenspaltennamen umsetzen.

find

```
←find dokTyp sgdTyp iqhandleT queryData
→foundCount
```

Suchen von Dokumenten anhand der vorgegebenen Suchkriterien. Die Treffer sind in die angegebene Hilfstabelle zu hinterlegen. Die Suchkriterien sind wieder als eine geschachtelte Liste zu übergeben:

```
{ColumnName cmpOp cmpValue} ...
```

Als Returnwert wird die Gesamtzahl der gefundenen Elemente, die in der Hilfstabelle enthalten sind, geliefert.

findAll

```
←findAll dokTyp sgdTyp iqhandleT
→foundCount
```

Suchen aller Dokumente anhand der Haupttabelle. Die Treffer sind in die angegebene Hilfstabelle zu hinterlegen. Als Returnwert wird die Gesamtzahl der gefundenen Elemente, die in der Hilfstabelle enthalten sind, geliefert.

delete

```
←delete dokTyp dokID
→delCntTab1 delCntTab2 ...
```

Es werden alle Datensätze, mit der angegebenen dokID in den beteiligten Tabellen gelöscht. Danach existieren für diese dokID keine Sachdaten mehr. Es werden die Anzahl der gelöschten Rekords pro Tabelle gemeldet.

get

```
←get dokTyp dokID skipRecNr
→{myColName columnValue} ...
```

Es werden alle Informationen, die in der DB für diese DokID vorhanden sind, ausgelesen und als Ergebnis geliefert. Sollte zu einer dokID mehrere Datensätze definiert sein, dann werden diese als Listen pro Spalte geliefert (analog zur add-Methode):

```
{myColName colValue1 colValue2 ...}
```

Es werden zuerst die allgemeinen Daten des Dokumentes und dann die *multi-ColumnValue* Listen geliefert. Die Anzahl der Datensätze bei den *multi-ColumnValue* Listen ist indirekt per [expr [llength $colInfo]-1] bestimmbar.

recache

←recache

→-empty-

Die Dokumenthandler-Skripte können eigenen Caches aufbauen um den Zugriff zu beschleunigen. Mit Hilfe dieser Anweisung sollten diese internen Caches reinitialisiert werden. Sollte keine Caches vorgehalten werden, dann ist trotzdem dies Methode bereitzustellen.

Dokumentzugriff per DokFileHandler-Prozedur

Bei der Definition von Dokumenttypen ist es möglich, daß der Zugriff der Dokumente auch über eine externe Quelle möglich ist. Hierzu muß ein sogenannter DokFileHandler definiert werden. Dieser Handler regelt die Kommunikation zur externen Datenquelle und lädt die angeforderten Dokumente in einen lokalen Cache, von dem aus der Server die Daten holen und an die Clients verteilen kann.

Der GeoServer liest aus der Dokumententypdefinition, die in der DB hinterlegt ist (Tabelle ***DoTypDefT***), welche Art der Dokumentenspeicherung vorliegt (hierzu dient der Eintrag ***TypStoreMode***)

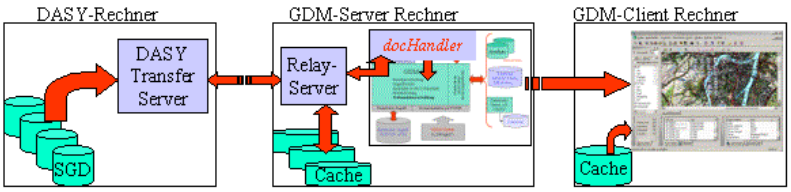
Derzeit sind die Modi:

0 → Speicherung im Dateisystem
1 → Zugriff per DokFileHandler
2 → Speicherung in DB als BLOB

definiert.

Im Modus 1 definiert der Eintrag "TypBlobInfo" den Namen der DokFileHandler-Prozedur, die die Kommunikation mit der externen Datenquelle abwickelt.

Üblicherweise wird zwischen der externen Datenquelle und dem DokFileHandler ein sogenannter RelayServer geschaltet, damit der GeoServer nicht "steht", wenn Dokumente von einer externen Datenquelle beschafft werden. Die Kommunikation zwischen den einzelnen Servern stellt sich dann wie folgt dar (Beispiel externe Datenquelle ist der Datenbestand aus dem DASY in Rheinland-Pfalz):



- Client fordert Dokument vom Server
- Server fragt per docHandler den Relay-Server ob Dokument vorhanden und eventuell auf aktuellem Stand
- Relay-Server fragt beim DASY-GDM Transfer-Server die Dokument-informationen ab:
  - ⇒ Prüfung ob Dokument im Cache noch aktuell
  - ⇒ falls vorhanden und aktuell, dann Freigabe an GDM-Server melden
  - ⇒ ansonsten Transfer der SGD-Datei zum Server und Umsetzung nach TIFF oder Umsetzung der SGD-Datei nach TIFF auf dem DASY-Rechner und Transfer der fertigen TIFF-Datei (je nach Einstellung in Steuerdatei)
  - ⇒ transferierte TIFF-Datei ins Cache ablegen und Freigabe an GDM-Server melden

Beim Hochfahren des GeoServers wird die Verbindung zur externen Datenquelle aktiviert. Je nach Definition der Komponente "`docHanderFailMode`", stoppt der GeoServer, wenn die Kommunikation nicht korrekt initialisiert werden kann (Wert `terminateServer`) oder läuft weiter, aber mit der Einschränkung, daß nun Dokumente nur aus dem Cache geliefert werden können (Wert `keepAlive`). Als Default wird für diese Komponente "`terminateServer`" verwendet.

Die einzelnen Methoden, die der DokFileHandler bereitstellen muß, sind die folgenden:

readDok

←`readDok dokTyp dokID`
→`waitTime`

Anfrage, das angegebene Dokument zu holen. Je nach Aktualität der Daten im Cache muß evtll. das Dokument von der Datenquelle geholt und vielleicht sogar noch in ein anderes Format gewandelt werden. Als Hinweis wie lange es dauert bis das Dokument bereitsteht, wird die Wartezeit in Millisekungen geschätzt.
Der GeoServer bzw. der Client muß danach wieder anfragen ob das Dokument verfügbar ist. Dies muß solange wiederholt werden, bis entweder ein Abbruch der Aktion gefordert wird, oder die Datei (endlich) vorhanden ist.

getGeoInfo

←`getGeoInfo dokTyp dokID`
→`dokTyp dokID x y ...`

Anfrage um für angegebenes Dokument, die Geoinformationen zu liefern.

getFileInfo

←getFileInfo dokTyp dokID
→dokTyp dokID Dateiname modTime fileSize lockedFlag
→time2Wait

Anfrage an den DokFileHandler die Dateiinformationen zum Dokument zu liefern. Dies wird insbesondere dann benötigt, wenn man einen Dateitransfer vom Server zum Client initiieren will. Sollte die Information nicht verfügbar sein, muß die Wartezeit geliefert werden.

getSimpleFile

←getSimpleFile $dokFilename $targetFilename
→waitTime

Anfrage, die angegebene Datei (ohne jegliche weitere Interpretation bzw. Formatumwandlung) von der Datenquelle zu holen. Als Antwort wird die Zeit in Millisekunden geliefert, die der Client bzw. der GeoServer warten soll, bis das Dokument vorhanden ist. Diese Wartezeit stellt eine statist. Schätzung dar.
Der GeoServer bzw. der Client muß diesen Aufruf sooft wiederholen, bis das Dokument verfügbar ist, in diesem Falle ist die Wartezeit 0.

initDocServer

←initDocServer dokTyp
→Versionsinfo vom dokFileHandler

Aufsetzen der Kommunikation zur externen Datenquelle. Es muß hier auch das Cacheverzeichnis abgefragt werden, so daß damit der Pfad zu den Dokumenten im Cache an der GeoServer gemeldet werden kann. Der Eintrag, wo die Dokumente zu finden sind, wird dann in dem Array

"dokTypSearchList($dokTyp)"

eingetragen.

initCache

←initCache
→Basispfad zum Cache

Analog zu pruneCache, jedoch wird mit diesem Befehl der Cache initialisiert, d.h. eventuell benötigte Verzeichnisse werden angelegt. Als Returnwert wird der Basispfad des Cache geliefert. Jeder Dokumententyp wird in einem eigenem Verzeichnis gespeichert.

pruneCache

←pruneCache
→aktuelle Größe des Cache in MB

Caches checken und eventuell aufräumen.

initGeoBase

←initGeoBase dokTyp
→gdmCommandFilename

Alle Dokumente, die im DatenServer zu diesem Zeitpunkt verhanden sind, abfragen und dafür alle Informationen (auch GeoInformationen) holen. Es wird wegen der zu erwartenden Menge an Daten hier eine

gdmCommand-Datei erstellt, die dann im GeoServer per *COMX*-Befehl ausgeführt wird. Die *gdmCommand*-Datei muß *DDOK* und *ADOK*-Anweisungen erzeugen bzw. beinhalten.

version

←`version`

→`Versionsinformation dokFileHandler und DatenServer`

Abfrage der aktuellen Versionen, diese Daten werden in die Log-Datei ausgegeben.

quitDocServer

←`quitDocServer`

→`-empty-`

Kommunikation zum DokFileHandler beenden. Der Handler kann dann runterfahren, d.h. seine Kommunikationskanäle schließen.

shutdown

←`shutdown`

→`-empty-`

Externen Dokhandler runterfahren, evtll. Shutdown an weitere Service-programme und/oder -server signalisieren.

tclCommand

←`tclCommand commandStr`

→`result`

Abschicken eines Tcl-Befehls an den DatenServer. Je nach Implementation des DatenServers, kann man duch Voranstellen eines "!"-Zeichen definieren, ob der Befehl im RelayServer oder im eigentlichen DatenServer ausgeführt wird. Dies setzt natürlich voraus, daß der DatenServer in Tcl realisiert ist, oder Tcl-Befehle irgendwie verarbeiten kann. Diese Methode kann auch eine Fehlermeldung generieren um damit diese Funktionalität abzuweisen.

deleteDok

←`deleteDok dokTyp dokID`

→`deleteCount`

Löschen eines Dokumentes im Cache und in der externen Datenquelle. Üblicherweise ist dieser Befehl nicht erlaubt und sollte eine entsprechende Fehlermeldung liefern.

writeDok

←`writeDok dokTyp DokID geoCoords dbDates dbData ↵`
`[hotSpot] [imgFormat]`

→`okFlag dokId`

Neues Dokument in externe Datenquelle einbringen. Dieser Befehl ist normallerweise nicht erlaubt, es sollte dann eine Fehlermeldung erzeugt werden.

getDokTypes

←`getDokTypes`
→`dokTyp ...`

Anfrage welche Dokumenttypen, dieser DokHandler verarbeiten kann.

Protokollierung von Benutzerzugriffen / Geschäftsbuch

In der Benutzerkennung wird hinterlegt, ob der Benutzer ein spezielle Login-Applikation zu druchlaufen hat, und es wird festgelegt, ob die Zugriffe (Lesen von Dokumenten, etc.) mitprotokolliert werden soll. Z.B. zum Führen eines Geschäftsbuches zur Abbrechnung.

Die zu aktivierende Login-Applikation und die Zugriffsprotokollprozedur ist in den `geoState`-Variablen **`applLoginCmd`** und **applCountXFR** hinterlegt. Überall dort, wo Dokumente gelesen werden, ruft der GeoServer diese eingestellte Prozedur mit einem entsprechenden Methodennamen und den angehängten notwendigen Daten auf.

Die einzelnen Methoden müssen die angeforderten Daten bzw. die bereitgestellten Daten aufbereiten und in die DB speichern bzw. von der DB holen (als Beispiel diene die Implementation von mhGeschBuch.tcl zur Realisierung eines Geschäftsbuches)

Es werden die folgenden Methoden vom GeoServer benötigt:

initGB

→ `""`
← `""`

muss zum (Re-)Initialisieren des Caches aktiviert werden

finalize4Logout

→ `""`
← `accept`

Der Benutzer hat sich ausgeloggt, es werden alle aktuellen Geschäftsbuch-/Protokolldaten, die im Cache sind, in die DB eingetragen.

newGByear

→ `year ?resetFlag?`
← `0/year`

neues Geschäftsjahr/Protokollierungsjahr festlegen. Wenn resetFlag gesetzt ist, kann auch das aktuelle Geschäftsjahr benutzt werden, dann wird die LfdNr auf 0 gesetzt (Vorsicht!).

getGByear
→ ""
← year
aktuelles Geschäftsjahr/Protokollierungsjahr erfragen.

getGBid
→ ""
← gbID
Es wird eine neue Geschäftsbuch- bzw. Protokollnummer vergeben und die DB mit einen Initialeintrag belegt.

cancelGBid
→ gbID
← gbID
Es wird die angegebene Geschäftsbuch- bzw. Protokollnummer gelöscht und falls dies die höchste Nummer war, dann wird lfd reduziert. Der eigene Eintrag kann ohne Privilegien gelöscht werden, alle anderen nur mit Admin-Recht.

currentGBid
→ ""
← gbID oder ""
Es wird die aktuelle Geschäftsbuch- bzw. Protokollnummer des Benutzers geliefert, sollte noch keine ID zugeordnet worden sein, dann wird ein Leerstring geliefert.

getGBdoks gbID
← gbID
→ {timeStamp dokTyp dokID} ...
Es werden alle Dokumente, die der Benutzer unter der angegebenen Geschäftsbuch-/Protokollnummer vom Server gelesen hat, geliefert. Diese Funktion ist nur dann sinnvoll, wenn die gelesenen Dokumente auch mitprotokolliert werden.

deleteGBentry
→ gbID
← gbID deletedCnt ...
Es werden alle Einträge in der DB unter dieser Geschäftsbuch-/Protokollnummer gelöscht. Der eigene Eintrag kann ohne Privilegien gelöscht werden, alle anderen nur mit Admin-Recht.

deleteUser uid
→ uid
← deletedCnt
Es werden alle Einträge in der DB gelöscht, die zu der angegebenen Benutzeridentifikation gehören. Nur mit Admin-Recht möglich. Dieser Befehl wird dann verwendet, wenn die Abrechnungsdaten nicht mehr verwendet werden (z.B. Benutzer wurde gelöscht)

getGBentry gbID

→ `gbID`

← `gbID {fname fvalue} {...} ...`

alle Daten des DB-Eintrags für die angegebene Geschäftsbuch-/Protokollnummer beschaffen.

setGBentry

→ `gbID`

← `gbID {fname fvalue} {...} ...`

schaltet auf eine bereits existierende Geschäftsbuch-/Protokollnummer um. Es werden nun alle Aktionen unter dieser ID protokolliert. Vorsicht, die ID muss existieren!

putGBentry

→ `gbID {fname fvalue} {...} ...`

← `gbID`

alle angegebenen Daten in der DB für die angegebene Geschäftsbuch-/Protokollnummer aktualisieren, dabei muß die gbID, die aktuelle vergebene Geschäftsbuchnummer sein (wie bei `getGBid` vergeben).

updateGBentry

→ `gbId {fname fvalue} {...} ...`

← `gbID`

alle angegebenen Daten in der DB für die angegebene Geschäftsbuch-/Protokollnummer aktualisieren, dabei kann dies jede (vorhandene) Geschäftsbuchnummer sein (die Berechtigung zum Ändern muß jedoch vorhanden sein).

findGBentry

→ `{fname cond fvalue} {...} ...`

← `queryhandle resultCount`

Suchen aller DB-Einträge, die die angegebenen Kriterien erfüllen. Als Ergebnis wird ein Handle geliefert, mit dem die eigentlichen Daten abgerufen werden können.

readGBentry

→ `queryhandle ?resetFlag?`

← `queryhandle gbID {fname fvalue} {...} ...`

← `queryhandle falls Ende der Daten`

Den nächsten Eintrag liefern, der bei der Suche via findGBentry gefunden wurde. Ist resetFlag ungleich Null, dann wird der Lesevorgang neu initialisiert und der 1. gefundene Eintrag wird geliefert.

closeGBentry

→ `queryhandle`

← `queryhandle`

Der Suchlauf ist abgeschlossen, es werden alle Daten wieder freigegeben.

recordDokID

→ `dokTypId dokID sizeCode hasGeoref`

← `""`

Eintragen eines vom Benutzer herangezogenen Dokumentes im aktuellen Geschäftsbuch-/Protokollierungsvorgang

deleteProtocolItem

→ `GByear lfdNr DokID`

← `NoDelete (kein solcher Eintrag vorhanden)`

← `OKNoReset (OK, aber kein ResetCounter)`

← `OK<size> (alles OK, auch Zähler wurden updated)`

löscht Eintrag in der Protokolldatei, indem ein und derselbe Eintrag mit einem - versehen noch einmal hinzugefügt wird. Es werden die Summenzähler entsprechend korrigiert (Spezialfall wg. MH-Applikation)

setProtocolItem

→ `oldGByear oldLfd dokID newGByear newLfd ?dokSize?`

← `NoEdit (kein Eintrag gefunden)`

← `NoCounterEdit`

← `OK<size> (alles OK)`

Ein Eintrag in mhGeschBDoksT wird einer neuen Geschäftsbuchnummer zugeordnet (Spezialfall wg. MH-Applikation).

clearGBlocks

→ `""`

← *`Liste aller entfernten Locks`* oder `"leer"`

Alle "Locks" auf Geschäftsbuchnummern werden entfernt

countProtocolRange

→ `GDMDate`

← `Anzahl Einträge seit angegebenem Zeitpunkt`

Liefert die Anzahl der Protokolleinträge seit einem vorgegebenen Zeitpunkt

deleteProtocolRange

→ `GDMDate`

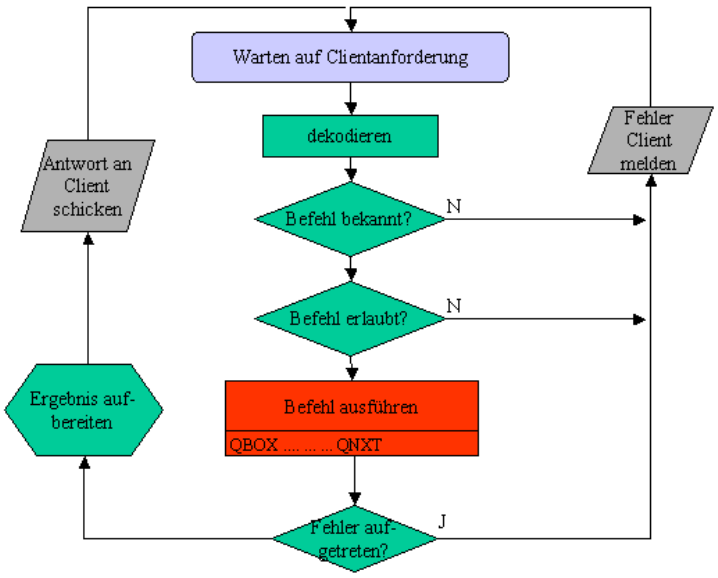
← `OK`

Löscht alle Protokolleinträge seit einem vorgegebenen Zeitpunkt

GeoServerBefehle

Der GeoServer ist ***passiv***, d.h. er wartet auf Anforderungen vom Clients. Diese Anforderungen werden eingelesen und analysiert. Hierbei wird der Verbindungszustand des Clients jeweils berücksichtigt, so daß in einem bestimmten Verbindungszustand nur bestimmte GeoServer-Befehle erlaubt sind.

Alle Aktionen, die der Client durchgeführt haben möchte sind auf diese Befehle zu reduzieren. Aus der Sicht des Servers sind alle Befehle gelichwertig, jedoch werden bei einigen Befehlen die Benutzerrechte geprüft. Die Unterscheidung *Einrichten, Auskunft, Update,* sind nur im Client definiert. Will man neue Befehle zum Befehlssatz des GeoServers hinzufügen, dann muß dabei die "Round-Trip"-Zeit beachtet werden, nämlich die Zeit, die benötigt wird um den Befehl vom Client zum Server zu schicken und bis zu dem Zeitpunkt, wenn die Antwort wieder beim Client eintrifft. Es sollten daher zeitaufwendige Aktionen vermieden oder in kleinere (Befehls)-Einheiten unterteilt werden, ist dies nicht möglich sollte man Hilfe der WAIT-Antwort den Client darüber informieren, daß sein Befehl angenommen und bearbeitet wird, jedoch langwierig ist und daher in Subeinheiten aufgeteilt wurde (z.B. beim Einsatz eine fileHandlers). Der Server schaltet nicht während der Abarbeitung eines Befehls auf einen anderen Client um, d.h. aus der Sicht eines Clients sind die Server-Befehle "atomar", d.h. es kann in dieser Zeit kein anderer Client die Abarbeitung bzw. die Daten beeinflussen. Der Befehlszyklus stellt sich wie folgt dar:



Die GeoServer-Befehle sind im folgenden aufgelistet, jedoch ohne den dazugehörigen Verbindungszustand. Neben diesen hier angegebenen Daten werden beim Datenaustausch von und zum Client noch weitere Daten transferiert, die aber nur zur Sicherstellung der Übertragung dienen.

In der Auflistung wird als `Request`, der Befehl aufgeführt, den der Client an den Server (inklusive zusätzlicher Daten) schickt und als `Reply` wird die Antwort des GeoServers auf den Befehl aufgeführt. Was diese Anweisung bedeutet, ist aus der Beschreibung zu ersehen. In der Beschreibung wird zuerst angegeben, welche Aktionsprozedur/-routine die Abarbeitung des Befehls übernimmt.

Client connect

Request:

Client connected zum GeoServer

Reply:

CONR GeoServerReleaseVersion GeoServerProtocolID SessionID sessionSeed ...

Beschreibung:

clientConnect --

Ein neuer GDM-Client möchte mit dem GeoServer Verbindung aufnehmen, es beginnt die "Login-Verhandlung". Wird die Verbindung akzptiert (d.h. der Verbindungsaufbau, noch nicht der eigentliche Login), dann sendet der GeoServer diese Antwort und erwartet als nächtes einen LGIN-Request.

Der GeoServer schickt Identifikationsinformationen an den Client, die dieser auswerten kann, insbesondere die aktuellen Versionen des Servers und des Verbindungsprotokolls

---

LGIN

Request:

LGIN user passwd

Reply:

CONN pwdExpireFlag loginApplFlag Berechtigung ...

Beschreibung:

actionLogin --

Clientverbindung wurde bereits akzeptiert, nun kommen die Login-Daten. Abprüfen der Login-Informationen mit den Daten in der DB. Ist alles ok, werden die Zugriffsberechtigungen geliefert und ein Flag ob eine zusätzliche Login-Applikation zu starten ist.

QUIT

Request:

QUIT

Reply:

QUIT

Beschreibung:

actionLogoff --

Client meldet sich vom System ab. Es wird die Verbindung beendet und die Login/Session-Informationen werden in der DB fortgeschrieben. Eventuell offene Anfragen etc. werden beendet.

CPWD

Request:

CPWD user passwd

Reply:

DWPC newExpireTime

Beschreibung:

actionChgPwd --

Client will Password ändern. Password übernehmen und entschlüsseln, das Password wird dann mit neuen Verfallszeitpunkt in die DB eingetragen. Das neue Verfallsdatum wird ebenfalls als Ergebnis geliefert.

ECHO

Request:

ECHO textstring

Reply:

OHCE textstring

Beschreibung:

actionEcho --

Client fordert einfaches Echo um Verbindung zu prüfen. Die Meldung wird in die Log-Datei geschrieben.

PING

Request:

PING

Reply:

GNIP {relVersion protocolID aliveSeconds}

Beschreibung:

actionPing --

Client schickt PING request, um so die Verbindung zu prüfen. Als Antwort wird die Serveridentifikation geschickt.

CERR

Request:

CERR msg

Reply:

-none-

Beschreibung:

actionError --

Client schickt Information, daß eine Serverantwort nicht korrekt war bzw. Client schickt eine zu registrierende Fehlermeldung, die der Server in seine Logs aufnehmen soll (z.B. um Sicherheitsverletzungen zu protokollieren)

SHUT

Request:

SHUT

Reply:

DONE {server shutdown} # an alle connected Clients

Beschreibung:

actionDone --

Server shutdown steht an, alle Clients benachrichtigen und anschließend Server veranlassen die Hauptarbeitsschleife zu verlassen.
Priviligierter Befehl!

RTCL

Request:

RTCL tclCommandString

Y Reply:

LCTR tclResultString

Beschreibung:

actionTclCmd --

Client schickt einen Aufforderung zur Abarbeitung eines TCL-Befehls. Der Tcl-Befehl wird im globalen Context ausgeführt. Dieser Befehl sollte nur zum Testen benutzt werden, ansonsten ist dieser Befehl nur auf einer sicheren Verbindung erlaubt (Admin-Berechtigung).

PARY

Request:

PARY arrayName [pattern]

Reply:

PARY listedArrayData

Beschreibung:
actionParray --

Client schickt einen Aufforderung zur Abarbeitung des parray-Befehl. Normalerweise würden die Daten dann auf der Console des Servers erscheinen. Mit diesem Befehl wird das Ergebnis von 'parray' in einer Liste gespeichert und an den Client zurückgegeben. Hier kann dann der Client die Ausgabe entsprechend ausführen (Admin-Berechtigung nötig).

INQS

Request:

INQS

Reply:

SQNI sgdFilename {xlu ylu xro yro} sgdMeridian [mainOvrFile]

Beschreibung:

actionInqSGD --

Client schickt Anforderung zur Lieferung der Abmessungen der Geo-Database (WorldBox) sowie des Namens der SGD-Datei. Es wird optional der Name der zugeordneten Übersichtskarte übergeben. Dieser Name wird nur dann geliefert, wenn der Client eine eigene Standardübersichtskarte im Login definiert hat, oder die Servereinstellungen eine spezielle Standardübersichtskarte definieren.

QPNT

Request:

QPNT xPos yPos

Reply:

```
TNPQ {xpos ypos} {utmzone utmeast utmnorth} \
   riss rk1000 xy2DMS xy2GEOREF
```

Beschreibung:
actionQryPnt --

Client schickt Anforderung um Informationen zu einem Punkt (in Weltkoord.) zu liefern. Eventuell wird der Punkt in das richtige Meridiansystem korrigiert. Es wird dann der Rißkode und der Blattnamen für M1:1000 geliefert. Als Zusatzinfo wird die Koordinate in geograph. Koord Länge/Breite in Form von Grad, Minuten, geliefert und der zugeordnete GEOREF-code geliefert.

QBOX

Request:

QBOX [-types typlist] [-like dokIDpattern] [-regexp] [-dateDok cond] [-dateLoad cond] Pnt ...

Reply:

XOBQ colhdl count

Beschreibung:

actionQryBox --

Es wird eine Selektion neu aufgesetzt, ist eine alte Selektion vorhanden, wird diese entfernt. Die Eingabekoordinaten werden in das RISS-SGD-System umgerechnet. Je nach Anzahl der Punkte wird eine Punktabfrage (mit entsprechender Defaultbox), eine Boxanfrage (Punkte: links unten, rechts oben) oder eine Selektion über ein Lasso durchgeführt.

Als Ergebnis wird die CollectionId und die Anzahl der gefundenen Elemente geliefert.

Wurde "-types" angegeben, dann werden nur Elemente dieses Dokumententyps bzw. dieser Dokumententypen selektiert. Wurde "-dokIDpattern" angegeben, dann werden nur die ID's selektiert, deren DokID diesen Ausdruck erfüllen. Ist "-regexp" angegeben, dann wird der Ausdruck in "-like" als ein regulärer Ausdruck (siehe regexp-Befehl in Tcl) angesehen, andernfalls ist dies ein "match-like" Ausdruck gemäß Informix inkl. MS-Access-Erweiterung und muß daher in einen regulären Ausdruck umgeformt werden. (MS-Access erlaubt [..] bzw. [!..] und interpretiert '#' als eine einzelne Ziffer).

Es gilt zu beachten, daß die DokIDs mit Blanks aufgefüllt werden, daher wird bei einem match-like Ausdruck nach der Umwandlung in einen regulären Ausdruck noch " *$" angehängt und ein "^" vorangestellt, d.h. der Ausdruck muß mit dem ganzen Eintrag übereinstimmen, im Gegensatz dazu werden bei einem regulärem Ausdruck (-regexp) keine Zusätze benutzt.

QNXT

Request:

QNXT colhdl [resetReadFlag]

Reply:

TXNQ colhdl isn {xpos ypos doktyp dokID {x1 y1 ...} [dokFilename]}
mappeFlag

Beschreibung:

actionQryGetNext --

Lesen des nächsten Eintrags aus der Collection (falls Flag nicht vorhanden, oder gleich 0), bzw. Lesevorgang neu beginnen und ersten Eintrag liefern (Flag ist auf non-Null gesetzt). Eigentlich wird intern die Collection aufbewahrt, aber falls mal mehrere Collections pro Client, dann ist die CollectionId (colhdl) der Startpunkt.
Ist Feature ConnectObjects geschaltet, dann erfolgt eine Prüfung, ob das Dokument Mitglied einer Mappe (entweder als Parent oder als Child) ist.

IQTL

Request:

IQTL ?-all|-recache?

Reply:

LTQI {dokTyp sgdTyp} {dokTyp sgdTyp} ...

Beschreibung:

actionInqTypList --

Client schickt Anforderung um Informationen über die registrierten Dokumenttypen zu erhalten. Es wird eine Liste der derzeit definierten Dokumenttypen, gemäß DB-Einträgen geliefert.

Ist die Option "-all" angegeben, dann werden alle Dokumenttypen, auch derzeit deaktivierte Dokumenttypen, geliefert.

Die Option "-recache" ermöglicht es, den GeoServer dazu zu veranlassen, daß alle Dokumenttypbeschreibungenneu von der DB eingelesen werden, danach werden alle aktiven Dokumenttypen geliefert.

IQTD

Request:

IQTD dokTyp

Reply:

DTQI dokTyp {id dokformat enabled georef ismap ismappe isfile isurkunde}
{dbTabOrProcName colcnt {colname coltyp colIsList colLabelString} ...}

Beschreibung:

actionInqTypDsc --

Client schickt Anforderung um Informationen über einen registrierten Dokumenttyp zu erhalten. Es wird der Tabellenname und die Tabellenbeschreibung(en) geliefert.
Die Tabellenbeschreibung wird in Form einer Liste geliefert, wobei der Typ eines DB-Feldes als GDM-Typ geliefert wird. Ist "colIsList" ungleich Null, dann ist dies eine Liste (d.h. es können zu diesem Feld viele Werte, max 32000, existieren). Die Beschriftung eines Feldes kann (sofern definiert) mit Hilfe des Textes in dem Element "colLabelString" durchgeführt werden. Für die Kommunikation mit dem GeoServer sind jedoch die Namen im Element "colname" zu benutzen.

IQGI

Request:

IQGI DokFilename
IQGI dokTyp dokID

Reply:

IGQI mercatorFlag {gdmWelt} {dateiWelt} {trafo} {imgwidth imgheight}

Beschreibung:

actionInqGeoWorld --

Client schickt Anforderung um Informationen über die Geoinformationen einer TIFF-Datei (WorldBox) in Meter zu erhalten. In der 2. Aufrufform, wird für das angegebene Dokument die Geoinformation beschafft.

Es wird in gdmWelt und dateiWelt die Boundingbox in der Form "xlu ylu xro yro" geliefert. Wobei "gdmWelt", die Koordinaten im GDM-Weltsystem darstellen (d.h. evtll. bereits umgerchnet wurden) und "dateiWelt" die Boundingbox im Weltsystem des Dokumentes darstellen.

Kann auch eine Transformationsmatrix bestimmt werden, bzw. ist eine Transformationsmatrix vorhanden, dann wird diese in der Form "a11 a12 a21 a22 tx ty" geliefert. Ist keine Transformationsmatrix verfügbar, wird die Einheitsmatrix "1 0 0 1 0 0" geliefert.

Mit imgwidth und imgheight, werden die Bildabmessungen in Pixel geliefert.

QYDB

Request:

QYDB sql-statement

Reply:

BDYQ sql-result...

Beschreibung:

actionQryDb --

Es wird eine SQL-Anweisung an die Datenbank weitergereicht und das Ergebnis geliefert. Diese Funktion ist nur zum Debuggen einsetzbar (Admin-Recht nötig)!

QBEG

Request:

QBEG [-types typlist] [-dateDok cond] [-dateLoad cond] [Pnt ...]

Reply:

GEBQ queryHandle

Beschreibung:

actionQryBegin --

wobei

typlist: "dokTyp dokTyp ..."

cond: "op date [op date]"

QADD

Request:

QADD queryHandle dokTyp {colName cond colValue [cond colValue]} ...

Reply:

DDAQ queryHandle

Beschreibung:

actionQryAdd --

@tbs

QRUN

Request:

QRUN queryHandle

Reply:

NURQ queryHandle count

Beschreibung:

actionQryRun --

Ausführen der bis dahin aufgebauten Suchanweisung. Die Suchanweisung wird damit geschlossen (d.h. es kann keine Erweiterung der Suchbedingungen per QADD mehr erfolgen) und die eigentliche Suche nach den Dokumenten wird durchgeführt. Es wird für diese Query eine temporäre Sachdatentabelle und evtll. eine Collection erzeugt. Anhand dieser Ergebnismengen, kann dann sequentiell jedes gefundene Dokument per QGET abgeholt werden.
Als Ergebnis wird die "geschätzte" Anzahl, d.h. die maximale Anzahl von möglichen Treffern geliefert. Die genaue Anzahl ergibt sich erst nach dem Auslesen aller Treffer, da die Geo-Einschränkung und die DB-Einschränkunen erst beim Holen eines Treffers kombiniert werden.

QDON

Request:

QDON

Reply:

NODQ completed

Beschreibung:

actionQryDone

Alle temporären Daten, Tabellen, etc., die zu der zuletzt durchgeführten Query gehören, löschen.

Request:

QDMP queryHandle

Reply:

BDYQ {dokTyp sgdTyp dokID} ...

TEGQ queryHandle -1 {nn nn dokTyp DokId {} [dokFilename]}

Beschreibung:

actionQryDump --

Alle Rekords aus der per QRUN fertig aufbereiteten Query-Zwischentabelle lesen und an Client übergeben. Vorsicht große Datenmengen...!

QGET

Request:

QGET queryHandle [ResetReadFlag]

Reply:

TEGQ queryHandle {dokTyp sgdTyp DokId}
TEGQ queryHandle => End of Cursordata

Beschreibung:

actionQryGet --

Den nächsten Treffer für die fertig aufbereitete Suchanfrage liefern (QBEG -> QADD ... -> QRUN -> QGET ...)
Sind keine Treffer mehr vorhanden, dann wird kein Trefferdatensatz geliefert. Wird der Flag ResetRead angegeben, dann erfolgt ein Zurücksetzen des Lesezeigers (Cursor) und die erste Treffer wird als Ergebnis geliefert. Mit ResetRead sollte man vorsichtig sein, da nicht alle Datenbanken eine Cursor-Reset erlauben, es muß dann der alte Cursor gelöscht und ein neuer aufgebaut werden.

ADOK

Request:

ADOK dokTyp dokID geoCoords dbDates dbData [hotspot] [imgFormat]

Reply:

KODA isn dokID fileNam

Beschreibung:

actionAddDok --

Vorbereitende Aktionen um ein Dokument in die Datenbasis aufzunehmen. Es finden die folgenden Aktionen statt:

- Prüfen ob Dokumententyp supported
- Prüfen ob Dokumententyp enabled
- Generieren einer DokID, sofern für DokTyp definiert
- Prüfen ob Dokument bereits registriert in DoGeoXTableT
- Eintragen in GeoBase, falls georef'd
- Eintragen in DoGeoXTableT, wobei dbDates zwei Datumswerte {scanDate dokDate} enthält.
- Eintragen der Sachdaten in der zugeordneten DB-Tabelle bzw. Eintrag per Skript-Methode, falls komplexe Sachdaten einzubringen sind.

Ist das Dokument keine Datei (nur DB-Relation und Flag isfile in DokTyp-Definition ist 0), dann ist der Transfer bzw. der Eintrag erledigt. Die wesentliche Komponente als Returnwert stellt dokID dar.

QDOK

Request:

QDOK [-what {xref|data|geo}] [-force] dokTyp dokID

Reply:

KODQ dokID xrefData dbData geoData lockID

Beschreibung:

actionQueryDok --

Fuer angegebenes Dokument sind die Geo und DB Daten zu liefern. Je nach Flags kann auch nur eine bestimmte Information angefordert werden. Ist für ein Dokument keine Geoinformation vorhanden, dann wird ein Fehler gemeldet (Dokument nicht vorhanden), mit dem Flag -force kann dieser Fehler unterdrückt werden, es wird dann versucht die angeforderte Informationen trotzdem zu liefern.

xrefData enthält:

ISN dokTypID dokID dokDatumSeq ladeDatumSeq
lockID refID refPntCount

dbData enthält den DB-Rekord (sofern vorhanden) in der Form {colName colValue} ...

geoData enthält die Geometriedaten, sofern das Dokument in der SGD enthalten ist. Format: ISN {xHotSpot yHotSpot xp1 yp1 xp2 yp2 ...}

Wird eine lockID != 0 gemeldet, dann bedeutet dies, dass die Daten einen Snapshot-Read darstellen, sie können sich ändern. Eine entsprechender Hinweis sollte beim Client erscheinen.

DDOK

Request:

DDOK ?-only {db|geo|file}? ?-force? ?-keepmappe? dokTyp dokID

Reply:

KODD countGeoDeleted CountDBDeleted CountFilesDeleted

Beschreibung:

actionDelDok --

Client fordert Server dazu auf, alle Dokumenteinträge unter der angegebenen dokID zu suchen und zu löschen. Es kann dabei vom Client angegeben werden, ob nur die Einträge in der DB und/oder in der Geodatenbasis zu löschen sind.

Ist `"-only file"` angegeben, dann wird nur das Dokument an sich gelöscht. Wird keine `"-only"`-Option angegeben, dann wird alles entfernt (also `-only "db file geo"`).

Es wird geprüft ob der Benutzer einen Lock auf das Dokument hat, bzw. ob ein anderer einen Lock definiert hat, wenn ein anderer einen Lock definiert hat, kann kein Löschen stattfinden. Sollte der Benutzer

keinen Lock definiert haben, dann wird ein temporärer Lock angefordert und am Schluß wieder freigegeben.
Ist "`-force`" angegeben, dann können auch ReadOnly-Dokumente gelöscht werden!
Normalerweise wird kann ein Dokument auch Mitglied einer Mappe sein, daher muß das Dokument aus der/den Mappe(n) gelöscht werden.
Ist jedoch "`-keepmappe`" angegeben, dann wird das Austragen des Dokumentes aus einer Mappe nicht durchgeführt.

RDOK

Request:

RDOK dokTyp dokID
RDOK -asfile dokFileName
RDOK -script scriptFilename
RDOK -module moduleFilename
RDOK -attach attchamentPathname (nur intern)

Reply:

KODR existFlag filePathOrEmpty handleOrEmpty fileSizeBytes fileModTime lockID
fileExtension

Beschreibung:

actionReadDok --

Client fordert Dokument an. Es soll entweder übertragen werden, oder eine Dateireferenz soll an den Client geschickt werden. Ist der

```
dokTyp == "-asfile",
```

wird die DokID als Dateiname interpretiert, z.B. der Name einer Übersichtskarte, die transferiert werden soll. Die Clientverbindung wird auf Modus Filetransfer vorbereitet (iohandle wird allokiert, die Datei wird geöffnet).
Ist `lockID != 0`, dann wird dieses Dokument von jemand bearbeitet und der Client kann anhand der lockID weitere Informationen abfragen.
Ist "`-script`" angegeben, dann wird die angegebene Datei in den Clientscript-Verzeichnissen gesucht.
Die Option "`-module`" wiederum führt dazu, daß die Datei in den Clientmodule-Verzeichnissen gesucht wird.

RDBL

Request:

RDBL rdhandle blockSize
XVBS rdhandle blockSize
XCLM rdhandle blockSize

Reply:

"binary Data from file to client"

Beschreibung:

actionReadBlock --

Client will nächsten Block vom Server lesen. Es wird der Client-Socket auf *'binary'* geschaltet, dann die angeforderten Daten transferiert und der Client-Socket wieder zurückgeschaltet. Wenn `'blockSize' <= 0`, dann Readvorgang beenden bzw. abbrechen.
Es gibt hier auch den Request *XVBS*, der Daten eines Skriptes anfordert. Der Lesevorgang wurde dann per *RVBS* aktiviert.

WDOK

Request:

WDOK -georef {{imgx imgy wrldx wrldy} ...} dokTyp dokID fileSize [fileExtension]

Reply:

KODW existFlag filePathOrEmpty handleOrEmpty

Beschreibung:

actionWriteDok --

Client möchte Dokument übertragen. Es wird alles für den Transfer vorbereitet und je nach Modus die Antwort geliefert. Ist lokaler Transfer enabled, dann ist der Parameter 'filePathOrEmpty' mit dem Namen der Datei definiert, die beim Kopieren als Zieldatei zu benutzen ist. Ansonsten ist filePathOrEmpty leer und handleOrEmpty enthält den IO-Handle, der bei den WriteBlock Messages zu benutzen ist.
Sollte das Dokument bereits vorhanden sein, dann wird die alte Datei auf {filePathOrEmpty}.bak umbenannt, dies ist bei einem fehlerhaften Transfer via lokalem Kopieren zu berücksichtigen, da dann ein Rename von .bak-Datei auf 'filePathOrEmpty' stattfinden muß. Ob eine alte Datei vorhanden ist, wird mit dem existFlag angezeigt.
Bei Option -georef werden Geoinformationen (Paßpunkte, die im Client definiert wurden) mit übernommen. Das Koordinatensystem ist links unten!
Der Dateityp (=Bild- bzw. Dokumentformat) kann zusätzlich übergeben werden, damit Dokumenttypen, die mehrere Bildformate erlauben, die Dokumente korrekt ablegen können (Workaround 'checkImageFormat' verwenden).

WRBL

Request:

WRBL wrthandle blockSize

Reply:

n/a startet mit Lesevorgang der binaeren Daten vom Client

Beschreibung:

actionWriteBlock --

Client will nächsten Block zum Server schicken. Es wird der Client-Socket auf 'binary' geschaltet, dann die angegebenen Daten

eingelesen und der Client-Socket wieder zurückgeschaltet. Wenn 'blockSize' <= 0, dann bedeutet dies Write-Vorgang beenden (=0, wg EOF) bzw. abbrechen (<0, wg. Fehler).

COMX

Request:

COMX subCmd subCmdData

Reply:

XOMC unknownSub → unbekannter Subbefehl
permDenied → keine Erlaubnis zum Ausführen
comandError sting → Fehler bei Ausführung Befehl
completed {daten ...} → erfolgreich abgearbeitet

Beschreibung:

actionExecuteMeta --

Client schickt Anweisung, die ausgeführt werden soll. Die Anweisung wird als SubCmd definiert, dem alle weiteren Daten folgen. Vor Ausführung des Befehls wird die Berechtigung zur Ausführung des betreffenden Unterbefehls geprüft. Dieser Befehl erwartet und liefert verschlüsselte Daten!

SubCmds:

`cmdFile >gdmCommandFile>`

Ausführen eines Commandfiles in dem Zeilenweise GDM-Protokollbefehle enthalten sind (Batch-Verarbeitung im Server). Es wird die angegebene Datei, die eine Folge von GeoServer-Befehlen enthalten, zeilenweise ausgeführt. Die Befehlsdatei mit GDM-Protokollbefehlen wird zeilenweise gelesen und jede Zeile als ein GDM-Befehl interpretiert, der auszuführen ist. Ab einem bestimmten Fehlerlimit wird die Verarbeitung abgebrochen. Es können in dem gdmCommandfile sogenannte Pragma-Anweisungen enthalten sein:

`#pragma silent`
Es werden keine Ergebniswerte an den Ergebnistext "resultString" angehängt.

`#pragma verbose`
Silent-Modus wieder abschalten. Nun werden Ergebnisse der einzelnen Befehle wieder an den Ergebnistext angehängt.

`#pragma continue`
Trotz eines aufgetretenen Fehlers weitermachen, d.h. es wird nicht nach einer maximalen Fehlerzahl die Verarbeitung abgebrochen.

`#pragma message`
Meldungstext als info-Meldung in die Logdatei schreiben.

Es werden die Ergebniswerte aller ausgeführten Befehle zusammengefaßt und am Schluß der Bearbeitung an den Client gesendet.

cmdString <gdmCommandString>

Ausführen eines GDM-Protokollbefehls. Es soll das angegebene Kommando "cmd" mit den zugehörigen Daten ausgeführt werden.

whoami -priv|-full

→ completed result...
Information über den eingeloggten Client liefern. Bei -full werden Benutzeradressinformationen mitgeliefert. Bei -priv werden die zugeordneten Benutzerrechte/-features geliefert.

applLoginBegin

→ completed accept
Während der Login-Phase des Clients, applikationsspezische Aktionen aktivieren (z.B. Geschäftsbuchsteuerung). Beginn der Aktionen definieren

applLoginCmd <subCmd> data...

→ completed result...
Während der Login-Phase des Clients, applikationsspezische Aktionen aktivieren (z.B. Geschäftsbuchsteuerung)

applRequest <applName> <subCmd> data...

→ completed ?result...?
Es wird eine applikationsspezische Aktion, die geoState(applList) hinterlegt ist, aktiviert. Der <applName> muß in der Liste der Applikationen vorhanden sein.

applLoginBegin

→ completed accept
Während der Login-Phase des Clients, applikationsspezische Akionen aktivieren (z.B. Geschäftsbuchsteuerung). Ende der Aktion anzeigen, nun weiter im Login-Vorgang.

applSql ?-noexec? "QYDB <sqlCmd>"

→ completed ?result...?
Applikationsspezifisches SQL-Statement ausführen lassen.

applArray <arName> <componentName>

→ completed ?result...?
Zugriff auf interne Daten des Servers ermöglichen (Vorsicht!). Ist das Array oder die Komponente nicht vorhanden, dann wird der Text "No match found" geliefert.

filename2PP <filename>

→ completed ?lux luy rux ruy rox roy lox loy?

Anforderung um aus einem Dateinamen, oder sonstiger Kennung die Paßpunktkoordinaten abzuleiten. Üblicherweise sind manche Kartennamen so aufgebaut, daß man daraus die Eckkoordinaten des Blattes ableiten kann. Es wird hier geprüft, ob im geoState-Array der Eintrag geoState(file2ppGenerator) vorhanden ist, wenn ja erfolgt die Umsetzung, ansonsten wird ein Leerstring geliefert. Konnte der Name vom Generator nicht umgesetzt werden, wird ebenfalls ein Leerstring geliefert.

RCYG

Request:

RCYG [-geodata] dokTyp dokID

Reply:

GYCR isn dokID pntCount
GYCR isn dokID pntCount pntData (bei -geodata)

Beschreibung:

actionRecoverGeoBase --

Recovery der SGD-Dateieinträge anhand der XRef-Daten in der DB. In der DB sind die Punkte sicherheitshalber hinterlegt, so daß eine Wiederherstellung der SGD-Daten möglich ist. Die neue ISN wird in der XRef-Tabelle nachgeführt.
Wurde -geodata angegeben, dann wird kein Recover durchgeführt, sondern es werden nur die Daten für ein Recover zurückgeliefert, insbesondere werden Lasso-Elemente aufgelöst.

LOCK

Request:

LOCK obtain|exclusiv <dokTyp> <dokID> <reasonString>
LOCK release <lockID>
LOCK release -all
LOCK convert <lockID> ?<exclFlag>?
LOCK cancel <lockID>
LOCK reset <lockID>
LOCK get <lockID>
LOCK info
LOCK islocked <dokTyp> <dokID>
LOCK check
LOCK find <query>
LOCK history <dokTyp> <dokID>
LOCK histInfo <lockID>

Reply:

KCOL lockID ...<je nach subCommand>

Beschreibung:

actionLocking --

Client fordert Sperrfunktionen an. Die Sperrfunktionen sind

obtain <dokTyp> <dokID> <reasonString>

für angegebenes Dokument eine Sperre wegen Änderungsarbeiten am Dokument setzen. Der Grund der Sperre muß im reasonString angegeben werden. Es kann das Dokument von anderen Benutzer zwar noch gelesen werden, aber kein anderer kann eine Änderung anfordern.

exclusiv <dokTyp> <dokID> <reasonString>

→ *lockID*

für angegebenes Dokument eine exklusive Sperre wegen Änderungsarbeiten am Dokument setzen. Diese Sperre führt dazu, daß das Dokument von anderen nicht mehr gelesen werden kann.

release <lockID>

→ *lockID fileModTime*

angegebene vorhandene Sperre freigeben. Als Returnwert wird der aktuelle Zeitstempel des Dokuments geliefert, damit der Client das Dokument im Cache entsprechend abgleichen kann.

release -all

→ *lockID ...*

alle vorhandenen Sperren des Benutzers, der diesen Request absetzt, freigeben.

convert <lockID> ?<exclFlag>?

→ *lockID*

konvertiert exisierende Sperre in den Modus "exclusive", wenn `exclFlag > 0`, oder nach "simple", wenn `exclFlag = 0`. Ist exclFlag nicht vorhanden, wird ein "convert lock mode" durchgeführt

cancel <lockID>

→ *lockID*

Eine Sperre zurücknehmen, ohne daß das Dokument verändert wurde (eventuell Restore der gesicherten Daten)

reset <lockID>

→ *lockID*

(nur mit Admin-Recht) Sperre eines anderen Benutzers aufheben.

get <lockID> ?<asMsgFlag>?

→ *lockID state dokTyp dokID user begin end reason*

Für angegebene Sperre, die zugehörigen Informationen liefern. Falls asMSgFlag vorhanden und ungleich Null, wird eine Standard-Meldung für die Sperre geliefert (keine vollständige Information, dafür aber im Klartext lesbar).

check

→ *lockID ... oder {}*

Prüfen ob für den Benutzer noch Sperren offen sind.

info

→ *lockID ... oder {}*

Synonym für "check"-Unterbefehl.

islocked <dokTyp> <dokID>

→*lockID oder 0 wenn kein Lock vorhanden*

Check ob angegebenes Dokument von irgendjemand gesperrt ist, wird 0 zurückgegeben, dann ist das Dokument nicht gelockt.

find <query>

→ *lockID ... oder {}*

(nur mit Admin-Recht) Auflisten der Sperren, die die angegebenen Kriterien (Liste mit colName cond colValue) erfüllen.

history <dokTyp> <dokID>

→ *lockID ... oder {}*

Anfordern der Lockhistorie des Dokumentes. Es werden alle Locks, die auf disem Dokument einmal definiert waren, als Liste von LockID's geliefert.

histInfo <lockID>

→ *user beginTime endTime reasonString*

Es werden zu einer LockID die Informationen wie Benutzer, der diesen Lock einmal definiert hat, wann der Lock begann und endete, sowie der Grund des Locks geliefert. Anhand der zeitlichen Reihenfolge kann dann Aufschluß darüber gegeben werden, was dieses Dokumente "erlebte".

ADMN

Request:

ADMN subCmd [args ...]

Reply:

NMDA value ...

Beschreibung:

actionAdmin --

Client schickt einen Aufforderung für Administrationsaktivitäten. Die Anforderung von Informationen, die nicht die Clientsession betreffen, erfordert Admin-Rechte. Folgende Subcommands sind definiert:

whoami ?-full|-priv?

Information über die eigen SessionID und UserID liefern. Ist -full angegeben, dann werden die Adressinformationen des Benutzers zusätzlich geliefert (in der Form "Realname PLZ Ort Strasse Telefon Fax"). Ist -priv angegeben, dann werden die Benutzerrechte/-features zusätzlich geliefert

listClnt

Alle Clients, die gerade mit dem GeoServer verbunden sind, auflisten. (Admin-Recht nötig)

inqClnt [sid]

Alle Informationen zum aktuellen Client, der diese Anforderung schickt, liefern. Ist eine Session-ID angegeben, dann werden Informationen zu dem angegebenen Client (sofern inzwischen die Verbindung nicht gelöst wurde) geliefert (Admin-Recht nötig)

shutClnt [sid]

Verbindung zum Client abbrechen. Dieser Befehl ist für die eigene Verbindung nicht sinnvoll, sondern vielmehr um andere Clients abzuschalten, hierzu ist deren Session-ID anzugeben (Admin-Recht nötig)

shutdownState [0|1]

Abfragen ob Server in Zustand "Pending Shutdown" bzw. den Zustand für Shutdown setzen oder zurücksetzen. (Admin-Recht nötig)

showlog [nlines]

Die letzten Zeilen (ca. 50-100 Zeilen, je nach Einstellung und Betriebssystem) der Log-Datei auflisten lassen. Manche Betriebssystem erlauben die Angabe wieviel Zeilen auszugeben sind. (Admin-Recht nötig)

checkPoint

Checkpointing auf die GeoBase-Datei durchführen. Dieser Vorgang kann nur mit Admin-Recht durchgeführt werden und dann nur, wenn kein anderer Client connected ist.

logonState [0|1]

Abfragen ob Logins erlaubt sind. Ist ein Parameter vorhanden, dann wird je nach Wert die Login-Möglichkeit ein- bzw. ausgeschaltet (nur mit Admin-Recht).

inactive ["check"|*newTimeOutValue*]

Aktuelle Einstellung der Inaktivitätszeit abfragen. Wird eine Zahl als Argument angegeben, dann wird diese Zeit (in Sekunden) als neuer Wert übernommen. Mit dem Argument "*check*" wird der Inaktivitätstest sofort ausgeführt, was dazu führen kann, daß inaktive Clients ausgeloggt werden. Als Returnwert wird der aktuelle Wert der Inaktivitätszeit geliefert.

accounts subCmd data...

Mit Administratorrecht (Ausnahme findUids und getEntry) Benutzerverwaltungsbefehle ausführen. Es sind die folgenden Befehle möglich:

addSimple userName ?address?

Es wird ein neuer Benutzer mit der angegebenen Kennung erzeugt. Der Benutzer erhält ein "Dummypassword", daß dann sofort zu ändern ist. Als Antwort wird die Benutzeridentifikation geliefert. Als wieterer Parameter kann die Adressinformation in der Form "RealerUsername PLZ Ort Strasse Telefon Fax". Wobei alle Anteile optional sind und als Leerstring angegeben werden können.

deleteUser ?-locks? ?-gbids? uname

Es wird der angegebene Benutzer gesucht aus der Benutzertabelle gelöscht. Mit den Optionen -locks und -gbids wird gesteuert, ob ebenfalls alle Einträge im Geschäftsbuch und den Locktabellen gelöscht werden sollen. Nur mit Admin-Recht ausführbar. Der Benutzereintrag wird in der Benutzertabelle zwar gelöscht, aber in der Tabelle in der gelöschten Benutzer übertragen, damit Referenzen auf diesen Benutzer noch aufgelöst werden können und ein historischer Nachweis geführt werden kann.

changePwd userName passwd

Als Superuser das Password eines anderen Benutzer ändern.

changeAddress userName {addrInfo...}

Ändern der Addressinformationen des angegebenen Benutzers. Die Addressinformationen sind als Feldliste anzugeben "{fieldName fieldValue} {fN2 fV2} ..."

enableState userName ?0|1? (Default 0)

Den Benutzer 'userName' deaktivieren bzw. reaktivieren (Argument=1). Dies ist dann z.B. nötig, wenn zuviele Login-Versuche stattfanden und der Benutzer deaktiviert wurde.

enableApplLogin userName ?0|1?

Benutzer 'userName' wird der Modus für die Aktivierung eines Applikationslogins geändert. Wird 1 angegeben, dann wird eine Applikation beim Login durchlaufen, die erst abgeschlossen werden muß, bevor der Benutzer die eigentliche GDM-Funktionalität nutzen darf. Ist kein Modus angegeben, dann wird der Defaultmodus aus dem Serverzustand geoState(applLoginMode) benutzt.

changeACL userName ACLname

Ändern der Zugriffserlaubnis. Es wird dem Benutzer die angegebene ACL zugeordnet.

availableACL

Liste aller ACL's, die derzeit in DB definiert sind, liefern "aclid aclname"...

resolveACL aclid

Umsetzen von AclID auf ACL-Name und den damit verbundenen Rechten. Mit dieser Information kann dann z.B. ein changeACL aktiviert werden.

resolveRights uid

→{privilege mode} ...

Für angegebene Userid die Liste der benutzerspezifischen Rechte liefern. Es wird neben dem Recht (=Privileg) auch angezeigt, ob diese Recht hinzugefügt (mode=0), oder entfernt werden soll (mode=1)

getEntry uid

Es werden alle Informationen des Benutzers mit der Kennung 'uid' geliefert. (Admin- bzw. applAdmin-Recht nötig)

findUids query

Alle Benutzerkennungen liefern, die die angegebene Anfrage erfüllen. (Admin- bzw. applAdmin-Recht nötig)

getAllEntry uid

Beschaffen der Daten für angegebene uid. Ist die uid nicht in der Benutzertabelle vorhanden, dann wird in der Tabelle der gelöschten Benutzer der historische Eintrag gesucht und als Ergebnis geliefert.

findDelUids query

Alle Benutzerkennungen liefern, die in der Tabelle der gelöschten Benutzer vorhanden sind und die angegebene Anfrage erfüllen (analog zu findUids).

insertTypeRestrictions uid changeList

Zugriffseinschränkungen auf bestimmte Dokumenttypen für angegebene Benutzerkennung eintragen. Die changeList ist eine Liste von Dokumenttypname und Zugriffskodenummer

help

Liste der möglichen Befehl liefern (nur mit Admin-Recht)

traceCmd [cmd [0|1]]

Befehlsverfolgung für Debugging ein-/ausschalten bzw. aktuelle Traceinformationen erfragen. Wird nichts weiter angegeben, dann werden alle Befehle, die eine Trace haben aufgelistet. Ansonsten kann ein einzelner Befehl gepüft bzw. der Tracemodus gesetzt werden.

typeRestrictions uid

Für angegebene Benutzerid die freigebenen Dokumenttypen, die dieser Benutzer auswählen darf, liefern. Es Dokumenttypen für den Benutzer eingeschränkt oder vollständig gesperrt sein.

Ist ein Dokumenttyp eingeschränkt, sind bestimmte Operationen auf diesem Typ nicht erlaubt.

CSET

Request:

CSET what value ...

Reply:

TESC oldValue

Beschreibung:

actionClntSet --

Client schickt Aufforderung eigene spezifische Daten/Einstellungen zu ändern (z.B. Sparache). Folgende Subcommands sind definiert:

language [eng|ger|...]

definieren der Sprache, in der Fehlermeldungen etc. an den Client geschickt werden. Ist ein Parameter angegeben, dann wird die neue Sprache eingestellt, die von nun an zu benutzen ist. Als Returnwert wird die "alte" Spracheinstellung des Clients geliefert.

help

Gibt Hilfeinformation zu möglichen Befehlen

IQFN

Request:

IQFN ?-path? dokTyp dokID

Reply:

NFQI dokTyp dokID filename fileSizeBytes fileModTime

NFQI dokTyp dokID {} 0; wenn Datei nicht vorhanden

Beschreibung:

actionBuildFilename --

Client fordert für Dokument den Dateinamen an. Es wird der Name der Datei (ohne Pfadinformationen) geliefert. Zusätzlich wird die Dateigröße und der Dateizeitstempel geliefert. Ist Flag -path angegeben, dann wird der vollständige (Cache-)Pfad der Datei geliefert.

QDTY

Request:

QDTY dokTyp

Reply:

YTDQ queryHandle dokcnt

Beschreibung:

actionQryDokTypAll --

Für einen bestimmten Dokumententyp alle Dokumente anhand der DB-Tabelle liefern. Ist dieser Dokumententyp ohne Sachdaten, dann wird eine Abfrage über die gesamte GeoBasis durchgeführt.

UDOK

Request:

UDOK dokTyp dokID geoCoords dbData hotspot

Reply:

KODU lockID changeDokID newDokID newDokFilename

Beschreibung:

actionUpdateDok --

Client will Update für ein bestehendes Dokument liefern. Es müssen nicht alle Elemente gefüllt sein. Um z.B. nur Sachdaten zu ändern, muß GeoCoords ein Leerstring bzw. {} sein. Das Dokument muß vorher gesperrt worden sein, bzw. die Sperre muß dem Benutzer gehören, der diesen Update durchführen will. Ändert sich durch den Update die DokID, dann wird dem Client mit dem Flag changeDokID==1 dies gemeldet, dann ist in newDokID die neue DokID und in newDokFilename der neue zugeordnete Dateiname.

QVBS

Request:

QVBS ?-sort?

Reply:

SBVQ {scriptFile mode} ...

Beschreibung:

actionQryScripts --

Anfrage vom Client, alle im Server definierten Clientskripte (z.B. VBS-Dateien) als Liste von Dateinamen und Zugriffsmodus zu uebermitteln. Ist Option -sort angegeben, dann wird die Liste nach den Scriptnamen sortiert. Der Zugriffsmodus ist ein Schlüsselwort mit der Bedeutung

`forceServer`

der Client muß das lokale Skript mit dem Server abgleichen und bei Unterschied wird die Datei beim Client überschrieben

`keepLocal`

ist die Datei am Client neueren Datums, als die Datei am Server, dann wird die lokale Datei am Client nicht überschrieben

---

RVBS

Request:

RVBS scriptfile

Reply:

siehe RDOK

Beschreibung:

actionReadScript --

Lesen eines Skript-Datei vom Server für den Client Es wird der Befehl umgelenkt auf RDOK. Dieser Befehl ist deshalb nötig, damit Skripte auch im Applikations-Login, zum Client transferiert werden können.

ENUD

Request:

ENUD subCmd subCmdData

Reply:

DUNE (Daten je nach Anforderung)

Beschreibung:

actionEnumDescr --

Client will Enumerationen handhaben, die in DB gespeichert sind. Die Aktion ist im SubCommand hinterlegt. Je nach Art des Befehl sind weitere Parameter nötig:

`list ?-full? ?pattern?`

→ cnt Name ...
→ cnt {id Name} ...
Auflisten aller Enumerationsnamen (oder ID plus Name, falls -full angegeben), die das Namenspattern erfüllen. Ist kein Muster angegeben, werden alle Enumerationen aufgelistet.

`get id`

→ {colName colValue} ...
Alle Daten für die angegebene Enumerationsid liefern

`find {colName cond colValue} ...`

→ id ... oder 0, wenn nichts gefunden
Suchen von Enumerationen anhand der angegebenen Bedingungen.

`enable ename ?onoff?`

→ enumID enableMode
Angegebene Enumeration disabled schalten (falls onoff nicht angegeben) oder je nach onoff-Wert schalten.

`help`

→ Liste der möglichen Subbefehle
Hilfe über mögliche Subbefehle liefern

ENUM

Request:

ENUM subCmd subCmdData

Reply:

MUNE (Daten je nach Anforderung)

Beschreibung:

actionEnumMemb --

Client will Enumerationen handhaben, die in DB gespeichert sind. Die Aktion ist im SubCommand hinterlegt. Je nach Art des Befehl sind weitere Parameter nötig:

`list ?-full? enumName`

→ cnt Name ...

→ cnt {id Name} ...

Auflisten aller Namen der angegebenen Enumeration (oder ID plus Name, falls -full angegeben)

`add enumName ememberName ?ememberID?`

→ {enumID ename} {emembID emembName}

Hinzufügen eines neuen Namens in die angegebene Enumeration. Ist ememberID angegeben, wird diese ID benutzt, sonst wird die nächsthöhere ID vergeben.

`remove enumName -id ememberID`

→ deleteCnt

Löschen eines Eintrags mit der Kennung ID in der angebenen Enumeration.

`remove enumName ememberName`

→ deleteCnt

Löschen des Eintrags mit dem angegebenen Namen aus der angegebenen Enumeration

`help`

→ Liste der möglichen Subbefehle

Hilfe über mögliche Subbefehle liefern

INFO

Request:

INFO

Reply:

OFNI serverID {Identifkation des Servers} {message of the day}

Die "Message of the day" erscheint nur wenn diese definiert ist.

Beschreibung:

actionInfo --

Client fordert schickt INFO request, um so die Serverkennung (ID und Serveridentifikationstext) zu erhalten. Ist in geoClient(n,infoMessage) ein Text vorhanden, wird dieser ebenfalls als "Message of the Day"

geschickt und anschließend diese benutzerspezifische Meldung gelöscht. Ist keine benutzerspezifische Meldung vorhanden, dann wird geprüft, ob eine allgemeine Meldung vorhanden ist, wenn ja, wird diese geliefert.

ATCH

Request:

ATCH <what> dokTyp dokID ...

Reply:

HCTA <data>

Beschreibung:

actionAttachment --

An einem Dokument kann ein Attachment (=Anhang) existieren in Form einer Zip-Archivdatei. In dem Archiv können mehrere Dateien enthalten sein, die Applikations oder Dokumenttyp abhängig sind, aber nicht vom Server weiter interpretiert werden. Attachments sind als Feature definiert und können abgeschaltet werden bzw. sein, d.h. es muß vorher ein Featuretest gemacht werden. Es sind die folgenden Unterbefehle definiert (nicht alle sind implementiert!):

Operationen auf Attachment an sich:

`exists dokTyp dokID`

→ 0 Kein Attachment vorhanden
→ 1 Attachment vorhanden
Prüfen ob das angegebene Dokument ein Attachment besitzt.

`list dokTyp dokID`

→ "" Kein Attachment vorhanden
→ {fileSizeBytes DateTime filePath} ...
Auflisten der Dateien, die im Attachment vorhanden sind. Das Dateidatum wird hierbei in der Form "dd-mm-yy hh:mm" übermittelt.

`get dokTyp dokID`

→ 0 {} {} 0 0 0 Attachment nicht vorhanden
→ analog zu RDOK-Returnwert: existFlag filePathOrEmpty handleOrEmpty fileSizeBytes fileModTime lockID
Das Attachment zu dem angegebenen Dokument zum Client übertragen. Es wird hier implizit ein RDOK-Befehl generiert. Danach kann das Attachment per RDBL blockweise gelesen werden.

`put dokTyp dokID fileSize`

→ analog zu WDOK-Returnwert: existFlag filePathOrEmpty handleOrEmpty
Client möchte ein komplettes Attachment, d.h. das Zip-Archiv zum Server übertragen. Es wird der Dateitransfer vorbereitet

und implizit ein WDOK-Befehl generiert. Danach kann das Attachment per WRBL blockweise transferiert werden.

delete dokTyp dokID

→ 0|1

Löschen des Attachments zum angegebenen Dokument.

update dokTyp dokID fileSize

→ 0|1

Client schickt ein Archiv mit Dateien. Der Server soll diese Dateien im Attachment des angegebenen Dokumentes aktualisieren und danach das Update-Archive löschen.

Operationen auf Dateien im Attachment:

extract dokTyp dokID fileName

→ 0 {} {} 0 0 0 Im Attachment angegebene Datei nicht vorhanden

→ analog zu RDOK-Returnwert

Client möchte aus Attachment nur eine bestimmte Datei lesen. Es wird die Datei extrahiert und an den Client ein Readhandle analog zu RDOK geliefert.

insert dokTyp dokID fileName fileSize

→ analog zu WDOK-Returnwert: existFlag filePathOrEmpty handleOrEmpty

Client möchte eine einzelne Datei übertragen und in das Attachment einfügen. Ist die Datei bereits im Attachment vorhanden, dann wird diese Datei überschrieben (existFlag wird entsprechend gemeldet).

remove dokTyp dokID fileName

→ 0|1

Client möchte eine einzelne Datei im Attachment des Dokumentes löschen. Ist die Datei nicht im Attachment vorhanden, dann wird 0 gemeldet, ansonsten 1.

replace dokTyp dokID fileName fileSize

→ analog zu WDOK-Returnwert: existFlag filePathOrEmpty handleOrEmpty

Client möchte eine einzelne Datei im Attachment des Dokumentes durch die angegebene Datei ersetzen. Ist die Datei nicht im Attachment vorhanden, dann wird diese Datei hinzugefügt (s. existsFlag)

FEAT

Request:

FEAT

Reply:

TAEF arrayValues

Beschreibung:

actionFeature --

Client schickt eine Aufforderung alle lizensierten bzw. aktivierten Features des Users/Servers aufzulisten.

QMAP

Request:

QMAP subCmd data...

Reply:

PAMQ result

Beschreibung:

actionMappe --

Client schickt einen Aufforderung zur Manipulation von Mappen. Eine Mappe ist eine Zusammenfassung von zusammengehörenden Dokumente in in hierarchischer Form. Die Dokumente müssen vorher eingebracht worden sein und können mit diesem Befehl verknüpft werden. Es sind hierbei die folgenden Befehle definiert:

`parent dokTyp dokID`

→ parentDokTyp parentDokID oder Leerstring

Liefert den Vater (Parent) zu diesem Dokument. Ist das Dokument kein Kind (Child) einer Mappe, dann wird ein Leerstring geliefert.

`children dokTyp dokID`

→ 0|n|-n

Anfrage wieviel Kinder (children) ein Mappendokument besitzt. Null wenn keine Kinder, sonst Anzahl der Kinder, wobei ab ein negativer Wert als Hinweis für den Client ist, daß die Anzahl zu groß ist, als das man die Child-Informationen per List-Subbefehl abholen kann, es sollte dann per beginchild usw. gearbeitet werden.

`list dokTyp dokID`

→ dokTyp dokID level ...

Listet alle Children des Dokumentes auf.

`beginchild dokTyp dokID`

→ rdhandle

Vorbereiten eines sequentiellen Auslesens aller Children einer Mappe.

`nextchild rdhandle`

→ childDokTyp childDokID level oder "" wenn Ende

Nächstes Child holen

donechild rdhandle

→ 0

Ende Lesevorgangs, der Readhandle wird wieder freigegeben.

add parentDokTyp parentDokID childDokTyp childDokID ?level?

→ 0|1

Eintragen eines neuen Childs in der angegebenen Mappe

remove parentDokTyp parentDokID childDokTyp childDokID ?level?

→ 0|1

Ändert das Child-Dokument dahingehend, daß es nun aus der Mappe herausgelöst wird. Das Child bleibt als eigenständiges Objekt erhalten.

delete parentDokTyp parentDokID childDokTyp childDokID ?level?

→ 0|1 (spätere Implementation)

Ändert das Child-Dokument dahingehend, daß es nun aus der Mappe herausgelöst wird. Das Child wird anschließend komplett gelöscht.

RCLM

Request:

RCLM ?-info? modulefile

Reply:

siehe RDOK

Beschreibung:

actionReadClientModule--

Lesen einer Module-Datei des GDM-Clients vom Server. Es wird der Befehl umgelenkt auf RDOK. Dieser Befehl ist deshalb nötig, damit aktualisierte GDM-Clientmodule auch im Applikations-Login, zum Client transferiert werden können (z.B. neue GDM-Clientprogrammversion oder aktualisierte DLL's).
Wird die Option `"-info"` angegeben, dann fordert der Client nur Informationen an, ohne daß irgendwelche Dateien zum Transfer vorbereitet werden, in diesem Falle wird als Antwort

KODR existFlag {} {} fileSizeBytes fileModTime 0 fileExtenstion

geliefert.

ICLM

Request:

ICLM ?-sort?

Reply:

MLCI moduleFilename ...

Beschreibung:

actionInqClientModules --

Anfrage vom Client, alle im Serverbereich definierten Clientmoduldateien (z.B. aktualisierte DLL's, neue GDM-Client-Programmversion usw.) als Liste von Dateinamen zu übermitteln. Bei "-sort" wird die Ergebnisliste nach den Dateinamen sortiert.

Der GDM-Client kann dann anhand der Informationsdatei "`GDM-Client.ini`" entscheiden, ob neue Module am Server vorhanden sind, die dann zum Client transferiert werden müssen.