# Functionele samenvatting

**Benodigde omgeving**

Op de lokale PC is een POSIX-BASH-shell nodig omdat de scripts die gebruikt worden daar van uitgaan. Zelf gebruik ik de MINGW64 die met github-client wordt meegeleverd.

De machine waar de postings worden geëxecuteerd is een machine van Marco Duiker. Dit is een Linux-machine. Er moet een SSH account worden aangevraagd, en daarin (in de root van de $HOME-directory) moet de git-repository van de xml\_validatietestcontent worden gecloond. Let op locatie zoals tussen haakjes weergegeven, want scripts gaan daar van uit.

De machine van Marco Duiker is speciaal omdat daarop een secure account richting LVBB bestaat, en omdat er een aantal ondersteunende Python-scripts staan die vanuit de BSH-scripts uit de repository worden gebruikt.

De git-repository wordt gebruikt door de lokale PC en de Linux-machine van Marco Duiker te laten communiceren. Immers alle test-datasets die op de lokale PC worden gecreëerd en worden gepusht staan daarna beschikbaar in de remote repository, en kunnen gepulld worden op de Linux-machine

**Naamgeving van data-set-directories**

Er mogen geen underscores (\_) in de naam voorkomen. Meestal is de naam als volgt:

{Naam soort validatie regel}/{nummer validatieregel}

Bijvoorbeeld LVBB1003

Soms zijn er meer validatie-regels op een nummer van toepassing, bijvoorbeeld bij LVBB1501, daarin is de beschrijving: “De datumBekendmaking binnen de opdracht is een datum, is in juiste formaat (JJJJ-MM-DD) zijn en mag niet in het verleden liggen.” Hier zijn twee test-datasets ingericht, een met een verkeerd formaat datum, en een met een datum die in het verleden ligt. Deze twee test-datasets heten: LVBB1501-1 en LVBB1501-2 (let op, geen underscores maar streepje als koppelteken)

**Benoemen nieuwe versie-directories:**

Indien er een nieuwe versie-directory moet worden aangemaakt heb ik de volgende policy:

Voorbeeld: 1.1.0-STOP-2.0.0-OW

Hierin staat de STOP-versie en de OW versie verwerkt. Ook het script om te genereren in de root van de repository wordt mutatis mutandis aangepast.

**Creëren één set validatie testdata**

Vanuit de directory gedacht die niet eindigt op generate-only

Ga naar de directory boven die waarin de xml’s van de testdata staan, geef volgende commando (voorbeeld):

../setscreate/unique-one.sh OZON0065

De geraakte en betrokken bestanden worden aangemeld op git, er is nog een “git commit” en een “git push” nodig om ze in de remote repository te krijgen

**Creëren alle sets validatie-testbestanden**

Ga naar de root van deze repository, run vanuit deze positie het shell-script:

generate\_all\_1.0.4-matrix-1.3-rc.sh

Indien het gaat om een gaat om een andere versie-directory, mutatis mutandis de namen aanpassen.

Aan het einde van dit shell-script wordt wel een “git commit” uitgevoerd, maar geen “git push”. Die moet nog worden gedaan.

Deze opdracht duurt ongeveer 2 uur (afhankelijk van machine)

**Executeren (posten) van één data-set**

Op de machine van Marco Duiker waarop de repository gecloond is, ga naar de root van de repository en dan naar de versie-directory (zoals hierboven bedoeld)

Een test-data-set is een zip-bestand en bevindt zich altijd in de subdirectory: {versie-directory}/opdracht\_voorbeeldbestanden/opdrachten\_gereed. Geef volgende commando:

execute\_tests/execute\_one\_without\_rebuild.sh {versie-directory}/opdracht\_voorbeeldbestanden/opdrachten\_gereed/data-bestand.zip

Het resultaat van de validatie is vanuit de root van de repository result/result.xml

**Executeren (posten) van alle data-sets**

Op de machine van Marco Duiker waarop de repository gecloond is, ga naar de root van de repository en dan naar de versie-directory (zoals hierboven bedoeld)

Geef volgende commando:

execute\_tests/execute\_all\_without\_rebuild.sh

De resultaten komen in result in een xml bestand, dat heet result09082021.xml (indien het op 9 augustus 2021 is gedaan)

**Resultaat bestand omzetten naar Excel bestand**

In de directory excel-export bevindt zich het xsl-bestand: create\_excel\_from\_result.xsl en errors\_index.xml. Het eenvoudigste om dit te gebruiken is door de result\*\*\*.xml van het test-resultaat in Oxygen te openen en dan te transformeren met deze xsl. Het resultaat is een Excel-sheet die in Excel kan worden geopend, en dan een mooie kleurrijke rapportage weergeeft voor niet technische gebruikers.

# Validatie-test-routines

Validatie-test routines zijn datasets waarin bewust een fout is aangebracht. Vervolgens wordt deze set ter validatie aangeboden aan de keten-software, om te kijken of de validaties op deze software deze fout vinden.

Om dit te doen is er een software-structuur opgezet die deze testen geautomatiseerd uitvoert.

Deze software voert verschillende handelingen uit

* Verandert de ID’s in de validatie-test-datasets, zodat de ketensoftware niet gaat zeggen dat er een bekende ID is aangeleverd. Soms wordt expres een ID herhaald, maar dan is dat onderdeel van de test. We gebruiken een speciale versie van de ID-Omkatter voor dit doel.
* Verandert datums in de data-sets, zodat die aan bepaalde (te testen) voorwaarden voldoet.
* Regelt dit allemaal via scripts
* Runt de bestanden op de machine van Marco Duiker die mag posten op de keten-software

Ik ga nu de directory structuur verklaren.

Het geheel bevindt zich in volgende github-repository: <https://github.com/Geonovum/xml_validatietestcontent>

In deze repository in de root zitten 4 directories en 2 shell-scripts om iets met die directories te doen. (De directories zijn elkaars spiegel, alleen in een andere versie. Normaal zitten er twee directories in.) Een heet OP-versie\_OW-versie, en de ander heet OP-versie\_OW-versie\_generate-only.

De tweede dient om de datasets te genereren, (dwz, datums en id’s veranderen) zodat je gelijk kunt doorwerken.

**generate\_all\_1.0.4-matrix-1.3-rc.sh**

Het genereren wordt gestart met het shell generate\_all\_1.0.4-matrix-1.3-rc.sh. Dat shell-script bereidt de “generate-only” directory voor op de actuele stand van zaken (kopieren van bron bestanden en genereren ID’s etc) en start een script om het genereren te starten, vervolgens doet het een git-commit maar geen git-push (omdat daarvoor een wachtwoord verschuldigd is)

**1.0.4-matrix-1.3-rc\_generate-only**

Deze directory wordt met rust gelaten. Hij wordt bijgehouden door bovengenoemd script en is niet bedoeld om in te werken. Alle handelingen in dit script worden zonder git gedaan, zodat er niet steeds nieuwe git-versies komen van de gewijzigde ID’s en datums. Dat zou vervuiling van git betekenen.

**1.0.4-matrix-1.3-rc**

Dit is de werk-directory. Hierin worden de bron-bestanden bewerkt. Deze dienen wel te worden opgegeven aan git. De scripts doen dit al.

**1.0.4-matrix-1.3-rc/opdracht\_voorbeeldbestanden**

In deze directory zitten de bron bestanden onderverdeeld naar validatie-type: BHKV, GEO, LVBB, OZON, STOP, TPOD. Daarnaast zitten er de directories AFBREEK, opdrachten\_gereed, setscreate en VERWIJDERD in. AFBREEK is een voorbeeld hoe een afbreek-routine te programmeren, heeft verder geen functie. In opdrachten\_gereed komen de gegenereerde test-datasets. In setscreate zitten enkele utility scripts. En in VERWIJDERD zitten validatie-test bestanden waarvan de validatie is uitgezet. Ik bewaar ze toch, want je weet maar nooit.

**1.0.4-matrix-1.3-rc/opdracht\_voorbeeldbestanden/BHKV**

**1.0.4-matrix-1.3-rc/opdracht\_voorbeeldbestanden/GEO**

**1.0.4-matrix-1.3-rc/opdracht\_voorbeeldbestanden/LVBB\_**

**1.0.4-matrix-1.3-rc/opdracht\_voorbeeldbestanden/OZON**

**1.0.4-matrix-1.3-rc/opdracht\_voorbeeldbestanden/STOP**

**1.0.4-matrix-1.3-rc/opdracht\_voorbeeldbestanden/TPOD**

Dit zijn de directories met de bron bestanden in subdirectories met een nummer.

Bijvoorbeeld **1.0.4-matrix-1.3-rc/opdracht\_voorbeeldbestanden/OZON/OZON0065**

In deze directory zit de dataset die validatieregel 0065 van OZON test. Men kan deze dataset gereed maken voor gebruik door in de directory **OZON**  te staan en dan volgend commando te geven:

**1.0.4-matrix-1.3-rc/opdracht\_voorbeeldbestanden/**

**../setscreate/unique-one.sh OZON0065**

De volgende uitvoer komt op het scherm:

*OZON0065*

*Script wordt uitgevoerd voor Validatie-testbestand*

*OZON0065*

*Trying to override old definition of task for*

*BUILD SUCCESSFUL*

*Total time: 30 seconds*

*/c/Users/Default.DESKTOP-NDDI22K/Documents/Geonovum/github/xml\_validatietestcontent/1.0.4-matrix-1.3-rc/ID-Omkatter/resultaat*

*creating /c/Users/Default.DESKTOP-NDDI22K/Documents/Geonovum/github/xml\_validatietestcontent/1.0.4-matrix-1.3-rc/opdracht\_voorbeeldbestanden/opdrachten\_gereed/opdr\_OZON0065-20210907114525-b-0-valideren.zip*

*adding: akn\_nl\_bill\_gm0297-3520-01.xml (164 bytes security) (deflated 82%)*

*adding: GoedeGebieden.gml (164 bytes security) (deflated 70%)*

*adding: GoedeGebieden.xml (164 bytes security) (deflated 70%)*

*adding: manifest.xml (164 bytes security) (deflated 82%)*

*adding: manifest-ow.xml (164 bytes security) (deflated 69%)*

*adding: opdracht.xml (164 bytes security) (deflated 38%)*

*adding: owLocaties-Gemeentestad.xml (164 bytes security) (deflated 84%)*

*adding: owRegelingsgebied.xml (164 bytes security) (deflated 67%)*

*adding: owRegeltekst-Gemeentestad.xml (164 bytes security) (deflated 74%)*

*adding: regelingsgebied.gml (164 bytes security) (deflated 58%)*

*adding: regelingsgebied.xml (164 bytes security) (deflated 71%)*

*git add /c/Users/Default.DESKTOP-NDDI22K/Documents/Geonovum/github/xml\_validatietestcontent/1.0.4-matrix-1.3-rc/opdracht\_voorbeeldbestanden/opdrachten\_gereed/oOZON006520210907114525b0v.zip;*

*/c/Users/Default.DESKTOP-NDDI22K/Documents/Geonovum/github/xml\_validatietestcontent/1.0.4-matrix-1.3-rc/ID-Omkatter/resultaat*

*git add /c/Users/Default.DESKTOP-NDDI22K/Documents/Geonovum/github/xml\_validatietestcontent/1.0.4-matrix-1.3-rc/opdracht\_voorbeeldbestanden/OZON/OZON0065/\**

Je kunt zien dat een build-script wordt uitgevoerd, vervolgens wordt alles aan git toegevoegd, ook de uitvoerbare dataset die bedoeld is om te posten richting LVBB. De oude dataset is verwijderd. De naam van de dataset is als volgt opgebouwd: o[validatienummer][datum/tijd] en het woordje b0v (valideren) of b0p (publiceren) of b0a (afbreken) of b1a (als tweede afbreken). Valideren geeft aan dat het een opdracht tot validatie is, b0 geeft aan dat het de 0 versie is. De b samen met de 0 staan voor de volgorde van afhandelingen. Soms bestaat een test uit meerdere data-sets, die dan ook op een bepaalde volgorde worden opgeruimd. De b0, of z0 of z1 zorgen hiervoor.

Als voorbeeld hiervan kijken we naar LVBB\_/LVBB4001, hiervan is de aangebrachte fout: “De aangeleverde AKN bestaat al : %1, publicatie niet mogelijk”

Hieruit wordt volgende bestand gemaakt

**1.0.4-matrix-1.3-rc/opdracht\_voorbeeldbestanden/LVBB\_/LVBB4001/bron**

**oLVBB400120210907124747a0p.zip:** Dit betekent dat we een publicatie moeten doen met een identifier. Om de publicatie achteraf te kunnen afbreken wordt deze werkzaam in de toekomst.

**1.0.4-matrix-1.3-rc/opdracht\_voorbeeldbestanden/LVBB\_/LVBB4001/bron\_1**

Hieruit wordt volgende bestand gemaakt

**oLVBB400120210907124747b0v.zip:** Daarna wordt een tweede dataset ter validatie aangeboden met dezelfde identifier. Deze is dus niet uniek en moet fout LVBB4001 triggeren

**1.0.4-matrix-1.3-rc/opdracht\_voorbeeldbestanden/LVBB\_/LVBB4001/bron\_afbreek**

Hieruit wordt volgende bestand gemaakt

**oLVBB400120210907124747z0a.zip:** En of dat gebeurt of niet, in ieder geval wordt de opdracht tot publicatie afgebroken

De meeste datasets die worden aangeboden zijn echter enkelvoudig en worden alleen ter validatie aangeboden en hoeven niet te worden afgebroken. (omdat ze geen publicatie bevatten)

**1.0.4-matrix-1.3-rc/opdracht\_voorbeeldbestanden/opdrachten\_gereed**

De bestanden die gereed zijn komen in deze directory.

Dan hebben we nog het script **create\_all\_sets.sh**

Dit script wordt opgestart vanuit de **1.0.4-matrix-1.3-rc/opdracht\_voorbeeldbestanden** directory.

Het voert de unique-one.sh uit met de benodigde parameter. Het is een simpel script en spreekt voor zichzelf.

**1.0.4-matrix-1.3-rc/ID-Omkatter**

Dit is het pièce de resistance. Hier gebeurt de magic. Ik heb voor Geonovum ook een ander ID-Omkatter gemaakt, deze is ervan afgeleid, maar houdt rekening met de bewust aangebrachte fouten.

Het wordt gestart vanuit unique-one.sh op regel 98 door: ant -q

Belangrijk in dit script zijn de bron en resultaat directories. De scripts zorgen ervoor dat dat goed komt, maar het is wel goed om het te begrijpen. Alle zaken die in een opdracht staan (met een ant-commando worden uitgevoerd) krijgen verwante identifiers. Het zijn gekoppelde data-sets, meestal betreft het een oorspronkelijke dataset plus een gewijzigde dataset. En afbreek-routines om publicaties af te breken, zoals uitgelegd in voorbeeld.

Het script zet de datasets in de bron-directories en haalt de resultaten op in de resultaat-directories. De temp-directory is voor intern gebruik.

Ik ga het toch uitleggen, deels als commentaar in de build.xml

**Build.xml**

Build.xml is een script dat wordt herkent door “ant” (installeren op PC en in pad opnemen)

Het heeft targets. Het enige target dat direct wordt aangeroepen staat onderin, het heet “main”

Vanuit “main” worden de andere targets aangeroepen

Build werkt in functie van ID-Omkatter

In de validatie-test is deze Omkatter complexer dan de pure Omkatter, immers hier moet hij niet allerlei bewust aangebrachte foutsituaties gaan repareren.

Laat ik stap voor stap beschrijven wat de ID Omkatter doet in de validatie-tester:

*Init*

Het init-target maakt een aantal variabelen aan die noodzakelijk zijn voor de overige targets. Dit gaat met name om de subdirectory namen. En het doet dit ook in sequence met een nummer erachter, zodat reeksen van validatie-sets goed gevalideerd kunnen worden

Tevens maakt de init ook de directories zelf aan.

En het maakt de variabele build.Currentdate aan die overal wordt gebruikt om identifiers te hernoemen. Deze wordt een keer aangemaakt zodat hij overal hetzelfde is.

*Copy*

In het copy target worden alle directories gevuld. Alles wat in de input (bron) directories staat wordt naar de output (resultaat) directories gekopieerd. Op deze wijze zijn we er zeker van dat bestanden die niet verder worden geraakt, toch in de output directories staan.

Tevens ondersteunt de ID-Omkatter twee afbreek directories, ook deze worden gevuld. Afbreek directories bevatten de bestanden die een publicatie kunnen afbreken.

*index*

De built.xml draait de index.xsl. In dit script worden een aantal XML-structuren opgezet die binnen de andere targets hergebruikt worden. Er wordt een index.xml aangemaakt voor iedere bron-directory. De index.xml wordt geplaceerd in de tem (of temp\_nummer) directory, en wordt niet meegenomen in het resultaat.

*Alle volgende Targets*

De variabele: org.file.dir ${env.orgfiledir} bevat de directory van de test set. Deze variabele is in het aanroepende script (unique-one.sh) aangemaakt. Vanwege deze environment variabele krijgen alle targets het property env mee.

Deze word gebruikt om identifiers op STOP-niveau om te vormen, zodat herkenbaar is welke test-set het betreft. Tevens wordt er het woordje AFBREEK in gezet indien er sprake is van een test-set die risico heeft om te blijven hangen op de keten-software (alle data-sets met AFBREEK in de identifiers kunnen worden weggegooid, ze horen er niet hen hadden eigenlijk door deze routines moeten worden weggegooid, maar dat is dan mislukt (misschien software-crash)

De alreadyRetrievedDateTime bevat de datum die in het init-target in ${build.currentDate} is aangemaakt en die ook wordt gebruikt om identifiers om te vormen.

*XSLT*

De XSLT routines zijn standaard routines met templates en functies. De functionaliteit is per soort target anders

De overige targets halen wat variabelen op in de index.xml. En geven deze variabelen door aan de xslt’s