

MSML Generic Parser – Manual

Table of Contents

1	Einleitung.....	2
2	User Guide.....	2
3	Aufbau des Parsers.....	2
3.1	Navigation.....	2
3.2	Parser-Instrumente.....	3
3.2.1	List-Extension.....	3
3.2.2	FileText-Extension.....	4
3.2.3	Limiter-Extension.....	5
3.2.4	Regex-Extension.....	6
3.2.5	Element Separator-Extension.....	6
3.2.6	Replace-Extension.....	7
3.2.7	Count-Extension.....	7
3.2.8	Fixed Text-Extension.....	7
3.3	Test-Textfeld.....	8
3.4	Menü.....	8
4	Kommandozeile.....	9
5	TODO.....	10
5.1	Pflicht.....	10
5.2	Nice-To-Have.....	10

1 Einleitung

Dieses Dokument soll eine Hilfe zur Erstellung von strukturierten MSML aus unstrukturierten Dateiformaten unter Verwendung des MSML-Parsers sein. Zuerst wird der Aufbau und die Benutzung des Programms erläutert und anschließend die Verwendung anhand von Beispielen. Zum Schluss werden einige noch zu bearbeitende oder nice-to-have Features erwähnt.

2 User Guide

Als erstes ist zu erwähnen, dass der MSML-Parser derzeit eine gewisse Vertrautheit mit dem MSML-Format voraussetzt. Sofern Zeit bleibt, kann der Parser zu einem einfacher zu bedienenden und mächtigerem Programm erweitert werden. In diesem Kapitel werde ich zuerst den grundlegenden Aufbau erklären und dann Details zu den Parser-Instrumenten.

3 Aufbau des Parsers

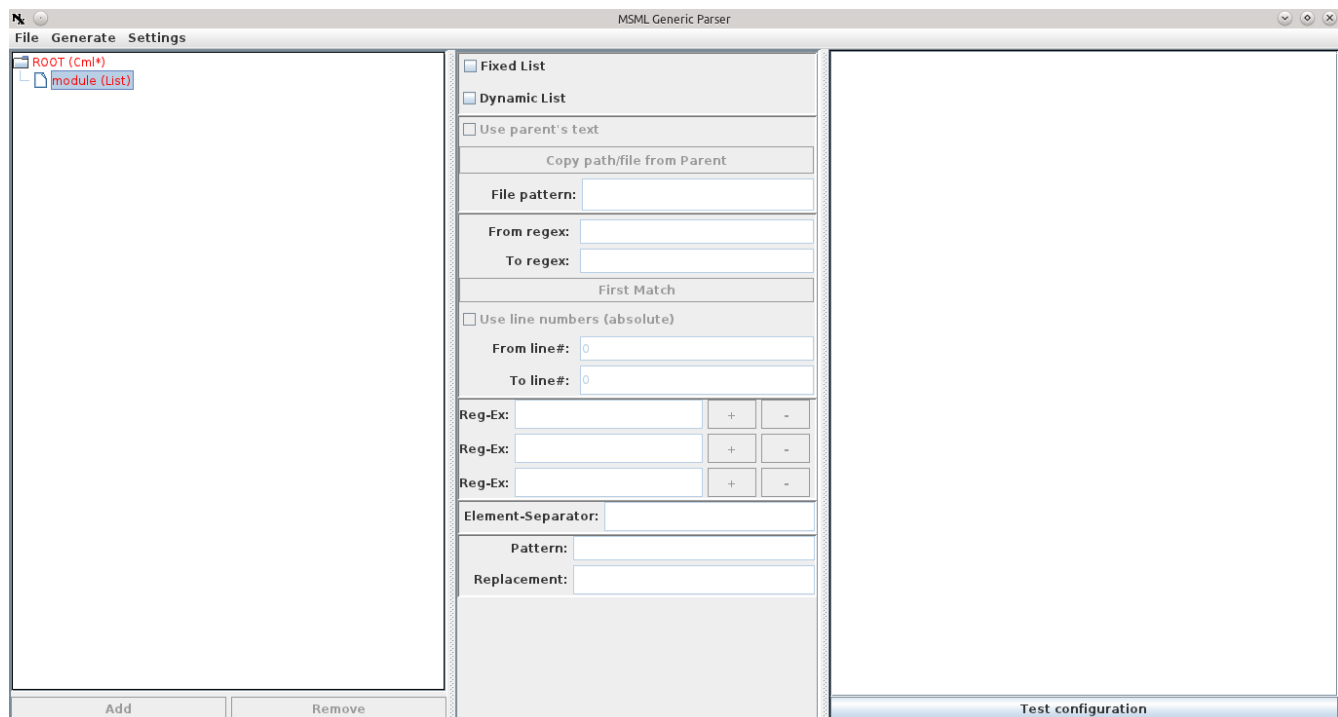


Abbildung 1: Parser GUI

Wie in Abbildung 1 zu sehen, gliedert sich der Aufbau des Parser in drei Teile. Links befindet sich ein Baum zur Navigation durch die MSML-Struktur. In der Mitte werden abhängig von der Auswahl des aktuellen Elementes in der Navigation die Parser-Instrumente angezeigt. Auf der rechten Seite befindet sich ein Textfeld zum Testen der aktuellen Einstellungen.

3.1 Navigation

Im Navigationsbereich können werden die einzelnen Elemente und Attribute eines MSML-Dokuments angezeigt.

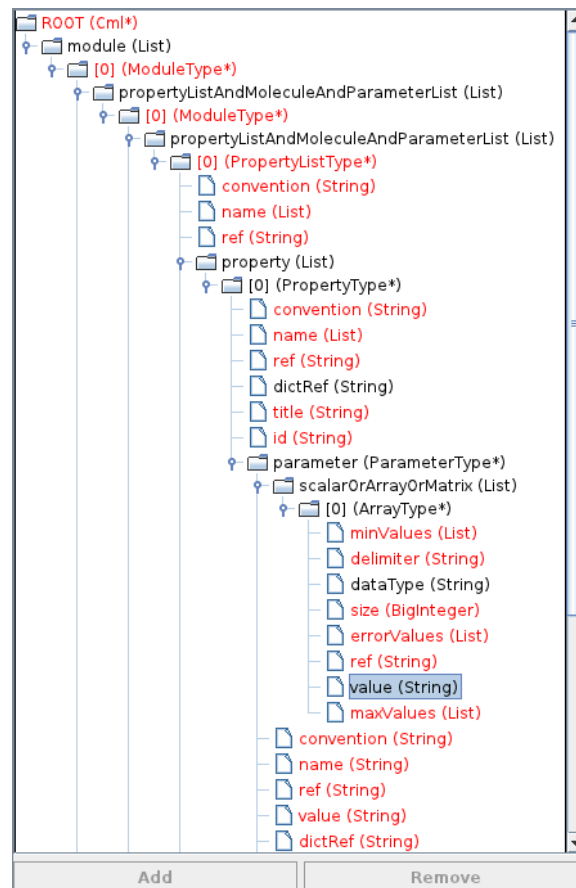


Abbildung 2: Navigations-Bereich

Die rot markierten Knoten in Abbildung 2 zeigen an, dass bisher keine Daten in die Parser-Instrumente eingetragen wurden. Diese Elemente und Attribute sind also leer. Für die schwarz markierten Elemente wurden Eingaben vorgenommen, so dass für diese Knoten auch MSML Elemente generiert werden. In der Klammer des jeweiligen Knotennamens steht der entsprechende Java-Typ, wobei MSML-Spezifische Typen zusätzlich mit einem Stern markiert werden.

Die Knöpfe Add und Remove sind nur dann aktiv, wenn im Navigationsbereich eine Liste ausgewählt wurde, die als „Fixed“ markiert wurde. Siehe dazu: List-Extension.

3.2 Parser-Instrumente

Der mittlere Bereich des Parser ist der wohl wichtigste, da hier alle Eingaben gemacht werden, um aus dem unstrukturierten Input den korrekten Wert zu extrahieren. Der Inhalt dieses Bereiches wird dynamisch an die Auswahl im Navigationsbereich angepasst. Im folgenden werden die einzelnen Instrumente erläutert.

3.2.1 List-Extension

Abbildung 3: List-Extension

Die List-Extension dient dazu dem Parser mitzuteilen, ob die Elemente der ausgewählten Liste

eine feste Anzahl haben oder ob die Größe abhängig von den Inputdaten ist. So enthält z.B. das erste Modul-Element in der Regel nur ein weiteres Modul-Element (die Joblist). Diese Liste hat somit eine feste Anzahl von Elementen. Ein Array von Atombindungen o.ä. in der Inputdatei hat jedoch in der Regel keine feste Anzahl von Elementen, sondern dies ist abhängig von dem berechneten Molekül. In diesem Fall würde man „Dynamic List“ auswählen. Siehe dazu auch Element Separator-Extension.

Wie in Abbildung 3 zu sehen, gibt es auch Situationen, wo „Dynamic List“ nicht auswählbar ist. Dies ist genau dann der Fall, wenn die aktive Liste sich aus unterschiedlichen Typen zusammensetzt. So enthält z.B. der Parameter eine Liste von Objekten, die entweder ein Scalar, Array oder Matrix sein können. In einem solchen Fall wird dem Benutzer nur die feste Liste zur Auswahl gestellt, der er Elemente durch die Knöpfe im Navigationsbereich hinzufügen und entfernen kann. Bei einer solchen List wird beim anhängen eines neuen Objektes dem Benutzer ein Auswahldialog zur Bestimmung des Types des hinzuzufügenden Elementes angezeigt.

Bei fixen Listen sind keine weiteren Eingaben zum Auffinden von Elementen dieser Liste notwendig – da der Benutzer ja Elemente manuell hinzufügt – so dass alle anderen Instrumente deaktiviert werden.

3.2.2 FileText-Extension



Abbildung 4: FileText-Extension

Die FileText-Extension ist die erste Erweiterung für alle nicht-Listenelemente. Diese Extension lädt den Rohtext, der den folgenden Erweiterungen zur Verfügung gestellt wird. Dies kann auf zwei Arten geschehen. Mit der Checkbox „Use parent's text“ wird der Text des Elternknotens als Grundlage für die folgenden Extraktionen verwendet. Dies ist besonders bei dynamischen Listen wichtig, da dann der „Elterntext“ der Text des dynamisch erzeugten Listenelementes ist.

Wurde die Checkbox aktiviert, dann ist die Textbox für das „File Pattern“ nicht mehr aktiviert. Diese Textbox stellt alternativ zum Elterntext die Möglichkeit zur Verfügung den Textinhalt einer Datei als Basis für die folgenden Extraktionen zu wählen. Hier gibt man einen regulären Ausdruck an, der dem zum aktuell eingestellten Basis-Verzeichnis relativen Dateinamen der gewünschten Datei entspricht. Es werden dabei nur Dateien berücksichtigt, die unterhalb des momentan eingestellten Basisverzeichnis oder in Subverzeichnissen zu finden sind. Zum Basisverzeichnis siehe: Menü. Der reguläre Ausdruck in „File pattern“ muss **genau einer** Datei entsprechen.

Beispiel: Nehmen wir an es existiert folgende Verzeichnisstruktur:

logs (Basis-Verzeichnis)

- temp
 - temp_23012012.log (a)
- log.log (b)
- temp_23012012.log (c)

Daraus ergeben sich also drei Dateinamen, die gegen den angegebenen regulären Ausdruck gemacht werden: „temp/temp_23012012.log“, „log.log“ und „temp_23012012.log“. Dabei ist zu beachten, dass bei Windowsplattformen der erste Dateiname statt „/“ ein „\“ als Verzeichnisseparator verwendet wird. Die folgenden regulären Ausdrücke würden, auf die entsprechenden Dateien matchen:

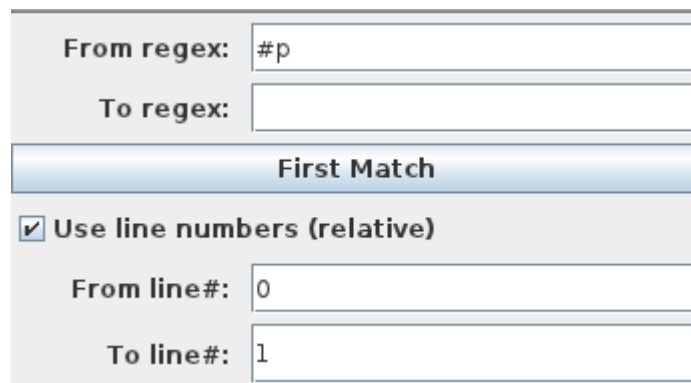
- a) `^temp[/\\]temp.*\.log$`
- b) `^log\.log$`**
- c) `^temp[^/\\]*\.log$`

Der reguläre Ausdruck in a) matcht alle Dateinamen, die mit einem „temp“ beginnen. In unserem Beispiel wären dies jedoch zwei Dateien, so dass der Ausdruck erweitert werden muss. Deshalb wurde ein Ausdruck hinzugefügt, der sowohl auf Linux, als auch auf Windowsplattformen dem Verzeichnisseparator entspricht. Zum Schluss wird nach dem eigentlichen Dateinamen gesucht, wobei die Datumszeichenkette durch das „.*“ abgebildet wird.

Im regulären Ausdruck für b) ist gut zu erkennen, dass der Punkt zwischen Dateinamen und Dateiendung mit einem Backslash „escaped“ werden muss.

Um den Dateinamen c) von a) zu unterscheiden, wurde der Verzeichnisseparator nach dem „temp“ explizit verboten, so dass nur die „temp“-Datei im Stammverzeichnis in Frage kommt.

3.2.3 Limiter-Extension



From regex:	#p
To regex:	
First Match	
<input checked="" type="checkbox"/> Use line numbers (relative)	
From line#:	0
To line#:	1

Abbildung 5: Limiter-Extension

Diese Extension erhält den Text direkt von der FileText-Extension und dient dazu den groben Text in einem ersten Schritt einzuschränken. Im oberen Teil dieser Extension können zwei reguläre Ausdrücke definiert werden und im unteren zwei Zeilennummern. Es lassen sich nun verschiedene Kombinationen bilden, wobei folgender Text an die nächste Komponente gereicht wird:

	„From regex“ definiert	„To regex“ definiert	Keine Regex definiert
„From line“ definiert	Text ab x-ter Zeile nach Vorkommen von „from regex“ bis zum Ende	Text ab der x-ten Zeile nach Vorkommen von „from regex“ bis zum Vorkommen von „to regex“ (TODO)	Text ab der x-ten Zeile des Dokuments bis zum Ende.
„To line“ definiert	Text ab x-ter Zeile nach Vorkommen von „from regex“ bis zur y-ten Zeile nach Vorkommen von „from regex“	Nicht anwendbar (TODO)	Text ab der x-ten Zeile des Dokuments bis zur y-ten Zeile des Dokuments
Keine Line definiert	Text ab der Zeile mit Vorkommen von „from regex“ bis zum Ende.	Text ab der Zeile mit Vorkommen von „from regex“ bis zur Zeile mit Vorkommen von „to regex“	Gesamter Text

Tabelle 1: Mögliche Kombinationen der Limiter-Extension

Mit dem Knopf „First Match“ kann bei der Suche nach den regulären Ausdrücken zwischen Vorwärts- und Rückwärtssuche gewechselt werden.

3.2.4 Regex-Extension

Reg-Ex:	<input type="text"/>	+	-
Reg-Ex:	<input type="text"/>	+	-
Reg-Ex:	<input type="text"/>	+	-

Abbildung 6: Regex-Extension

Diese Extension erlaubt es den noch verfügbaren Text anhand mehrerer regulärer Ausdrücke weiter einzuschränken. Diese Ausdrücke werden nacheinander auf den jeweils noch übrigen Text angewendet, wobei die Pattern im Multiline-Modus kompiliert sind. Dadurch wird eine Suche nach einem Match, der über mehrere Zeilen geht, ermöglicht.

3.2.5 Element Separator-Extension

Element-Separator:	<input type="text"/>
--------------------	----------------------

Abbildung 7: Element Separator-Extension

Diese Erweiterung ist nur für Listen sichtbar und wie die anderen nur aktiv, wenn es sich um eine dynamische Liste handelt. In dieser Textbox wird ein regulärer Ausdruck definiert, der die Elemente einer dynamischen Liste aus dem Grundtext trennt. Für jedes getrennte Element wird dann ein Element der dynamischen Liste angelegt und diese können sich über die FileText-Extension durch Aktivierung der Checkbox „Parent's Text“ den entsprechenden Text holen.

3.2.6 Replace-Extension

Pattern:	<code>^,.*#(.*)\n</code>
Replacement:	<code>\$1</code>

Abbildung 8: Replace-Extension

Diese Erweiterung erlaubt es auf dem verbliebenen Text eine Substitution durchzuführen. Dabei wird nach allen Vorkommen des in „Pattern“ angegebenen Musters gesucht und durch den in „Replacement“ angegebenen Ausdruck ersetzt. Dabei ist es möglich mit \$1... \$n auf die in „Pattern“ durch runde Klammern definierten Gruppen zuzugreifen. Im Beispiel aus Abbildung 8 wird also jede verbliebene Zeile, in der eine Raute vorkommt, durch die Zeichen zwischen der Raute und dem Zeilenende ersetzt.

3.2.7 Count-Extension

Count Pattern:	<code>\s</code>
-----------------------	-----------------

Abbildung 9: Count-Extension

Diese Extension zählt das Vorkommen des im Textfeld angegebenen regulären Ausdrucks und liefert das Ergebnis als Text zurück. So wird z.b. der Text „aabbcc“ durch den regulären Ausdruck „[^b]“ in den Text „4“ übersetzt.

3.2.8 Fixed Text-Extension

Fixed Text:	
--------------------	--

Abbildung 10: Fixed Text-Extension

Diese Extension ignoriert alle vorangegangenen Verarbeitungsschritte und setzt den Ergebnistext auf den im Textfeld definierten fest.

3.3 Test-Textfeld

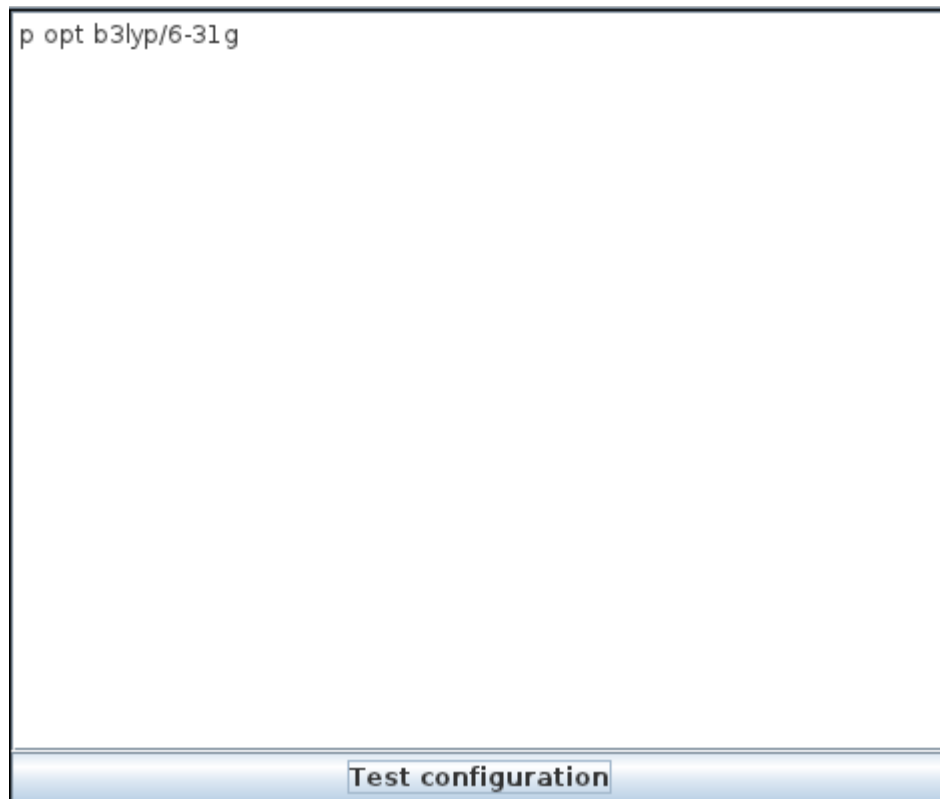


Abbildung 11: Test-Textfeld

Das Textfeld auf der rechten Seite des Parsers dient zur Überprüfung der Einstellungen für das aktuell ausgewählte Element. Mit der Betätigung des „Test configuration“-Knopfes wird der gleiche Prozess angestoßen, der bei der vollständigen Erstellung eines MSML-Dokumentes für jedes nicht-leere Element der Konfiguration läuft. Der Parser liest den Basistext (Elterntext oder Text der gefundenen Datei), wendet alle gültigen Extensions an und gibt den verbliebenen Text im Textfeld aus. Dies ist dann genau der Wert, der in einem MSML-Dokument dem momentan ausgewählten Element oder Attribut zugewiesen wird. Dabei wird beim Testen der Konfiguration nicht überprüft, ob der Wert selbst gültig für das Element ist.

3.4 Menü

Im Menü finden sich folgende Optionen:

- File:
 - Save Configuration: Speichern die aktuelle Konfiguration als XML an einem zu wählenden Ort ab.
 - Load Configuration: Lädt eine zu wählende Konfiguration in den Parser.
 - Exit: Beendet die Anwendung.
- Generate:
 - MSML: Generiert ein MSML-Dokument aus den aktuellen Daten und speichert dieses an einem zu wählenden Ort ab. Dies dient zur Überprüfung aller Einstellungen.
- Settings:

- Set base path: Mit dieser Menüoption wird der Basispfad ausgewählt, unter dem die zu verarbeitenden unstrukturierten Dateien liegen. Der Parser betrachtet beim Vergleich des regulären Ausdrucks für die Dateinamen (siehe FileText-Extension) nur Dateien, die in dem gewählten Ordner oder einem Subordner liegen.

4 Kommandozeile

Wird der Parser ohne Kommandozeilenargumente gestartet, so wird das im vorangegangenen Kapitel erläuterte grafische Interface gestartet. Dieser Modus wird benutzt, um Konfigurationsdateien für den Parser zu erstellen. Der Parser hat jedoch weitere Aufgaben, für die Kommandozeilenparameter verwendet werden.

Es ist möglich eine bestehende MSML-Datei zu laden und diese um einen geparsten final- oder calculation-Block zu erweitern. Dies ist die Hauptverwendung des Parsers im laufenden Betrieb. Dabei werden dem Parser Parameter in der Kommandozeile übergeben, die auf eine MSML-Datei verweisen, eine JobID enthalten und das Basisverzeichnis angeben. Der Parser lädt dann die angegebene MSML-Datei und sucht den der JobID entsprechenden job. Dieser Job kann dann verschiedene Parameter enthalten, die im Parser-Dictionary erläutert werden.

(TODO) Zudem soll es dem Nutzer des Mosgrid-Portlets möglich sein verschiedene Konfigurationsdateien auszuwählen, die nur eine Eigenschaft aus den Outputdateien extrahieren. Diese Daten sollen dann in den letzten final- oder calculation-Block eingefügt werden.

```
USAGE: genericparser [-f <FILE> -j <JOBID>] -d <DIRECTORY>
-f : MSML-File that contains the <JOBID> that has to be processed.
-j : JobID contained in the MSML-File.
-d : Base-Directory where the parser looks for the output-files.
```

Optionen:

-f <FILE>:

Dieser Parameter bezeichnet die MSML-Datei, die geladen werden soll. Diese Datei enthält Informationen, wie der Parser die Informationen aus dem Basisverzeichnis extrahieren und der MSML-Datei hinzufügen soll.

-j <JOBID>:

Dieser Parameter muss mit einem der JobIDs in der gegebenen MSML-Datei übereinstimmen. Der Parser verarbeitet dann den entsprechenden job.

-d <FILE>:

Diese Option gibt das Basisverzeichnis an, wo alle Outputdaten des zu verarbeitenden Tools zu finden sind.

5 TODO

5.1 Pflicht

- Fixed-Text-Extension sollte alle anderen Extensions deaktivieren und leeren.
- Limiter-Extension „to regex“ Kombinationen müssen überarbeitet werden.

- Mergefunktionen
- Count-Extension, die das Vorkommen eines regulären Ausdrucks zählt und als Text zurückgibt.
- Namespaces für dictRef und datatype attribute müssen auf beim Mergen mit den im gegebenen MSML-Dokument vorhandenen namespaces abgeglichen werden, damit die Prefixes korrekt gesetzt werden.

5.2 *Nice-To-Have*

- Element-Separator muss mandatory sein bei dynamischen Listen.
- Test configuration sollte überprüfen, ob der Wert nach JAVA gültig für das Element ist.
- Comboboxen mit dictRef und datatype Auswahlmöglichkeiten.