Rem

RemRes Protkoll

RemRes XML-basiertes Protokoll

Christoph Hauer

Version 0.2

FH Wiener Neustadt

BIT11

Inhalt

[1 Grundbausteine 2](#_Toc372804537)

[1.1 WatchField ohne Werte 2](#_Toc372804538)

[1.2 SingleValue 2](#_Toc372804539)

[1.3 WatchValue 2](#_Toc372804540)

[1.4 WatchField mit ermittelten Werten 2](#_Toc372804541)

[1.5 WatchRule 3](#_Toc372804542)

[1.5.1 WatchParameter 3](#_Toc372804543)

[1.5.2 Attribute 3](#_Toc372804544)

[1.6 OperationStatus 4](#_Toc372804545)

[1.7 Process 4](#_Toc372804546)

[1.8 Setting 4](#_Toc372804547)

[2 Messages 5](#_Toc372804548)

[2.1 AddWatchRule 5](#_Toc372804549)

[2.2 NotifyMe 5](#_Toc372804550)

[2.3 ClearWatchRules 5](#_Toc372804551)

[2.4 DeleteWatchRule 5](#_Toc372804552)

[2.5 GetWatchData 5](#_Toc372804553)

[2.5.1 All Watched Data 5](#_Toc372804554)

[2.5.2 For a specified WatchField 6](#_Toc372804555)

[2.5.3 For a specified WatchRule 6](#_Toc372804556)

[2.5.4 Result 6](#_Toc372804557)

[2.6 GetWatchRules 7](#_Toc372804558)

[2.6.1 Result 7](#_Toc372804559)

[2.7 Notification 7](#_Toc372804560)

[2.8 Keep Alive 8](#_Toc372804561)

[2.8.1 Keep Alive Request 8](#_Toc372804562)

[2.8.2 Keep Alive Response (Result) 8](#_Toc372804563)

[2.9 GetProcessList 8](#_Toc372804564)

[2.9.1 Result 8](#_Toc372804565)

[2.10 Settings 9](#_Toc372804566)

[2.10.1 GetSetting 9](#_Toc372804567)

[2.10.2 SetSetting 9](#_Toc372804568)

[3 Glossar 9](#_Toc372804569)

# Grundbausteine

## WatchField ohne Werte

Das **WatchField** Element ohne Werte beschreit ein Element des Computer das Überwacht werden soll. Die Werte **WatchObject** und **WatchProperty** können sich auf das WMI System von Windows beziehen aber auch frei definiert werden. Das WatchField darf immer nur ein Object als Result liefern. Dies kann mittels der **WhereClause** sicher gestellt werden. Das **WhereClause** Feld ist optional.

<WatchField Type=” *#Types#* ”>

<WatchObject> *#WMI Object#* </WatchObject>

<WatchProperty> *#WMI Object Property#* </WatchProperty>

<WhereClause> *#WMI Where Clause#* </WhereClause>

</WatchField>

*#Types# =* String (Default), Integer, DateTime, Double, Boolean

Dieser Typ wird verwendet um es zu ermöglichen beim Vergleich in einer WatchRule richtig auswerten zu können.

## SingleValue

Das **SingleValue** wird dazu verwendet um fixe Werte anzugeben.

<SingeValue Type=” *#Types#* ”>

<Value> *#Ein fixer Wert#* </Value>

</SingleValue>

## WatchValue

Ein **WatchValue** Element ist ein überwachter Wert eines **WatchField** Elements zu eine definierten Zeit mit einem gewissen Ausprägungswert.

<WatchValue>

<Value> *#Ein Ausprägungswert Wert#* </Value>

<MomentOfMeasure>

*#Messzeitpunkt# {Format: dd.mm.yyyy HH:mm}*

</MomentOfMeasure>

</WatchValue>

## WatchField mit ermittelten Werten

Das **WatchField** Element mit ermittelten Werten wird dazu verwendet um Daten von Service zu zurückzugeben.

<WatchField Type=“ *#Types#* “ >

<WatchObject> *#WMI Object#* </WatchObject>

<WatchProperty> *#WMI Object Property#* </WatchProperty>

<WhereClause> *#WMI Where Clause#* </WhereClause>

<WatchFieldValues>

<WatchValue>

<Value> *#Ein Ausprägungswert Wert#* </Value>

<MomentOfMeasure>

*#Messzeitpunkt# {Format: dd.mm.yyyy HH:mm}*

</MomentOfMeasure>

</WatchValue>

***{0 ... n}***

</WatchFieldValues>

</WatchField>

## WatchRule

Das **WatchRule** Element wird dazu verwendet um eine Überwachung für einen Ausprägungswert zu definieren.

<WatchRule Name=”*#Name of Rule#*” Notify=”*#true/false#*”

Period=”*#Ueberwachungsperiode#*” CacheValues=”*#AnzahlValues#*” >

<WatchField Type=” *#Types#* ”>

<WatchObject> *#WMI Object#* </WatchObject>

<WatchProperty> *#WMI Object Property#* </WatchProperty>

<WhereClause> *#WMI Where Clause#* </WhereClause>

</WatchField>

<WatchParameter>

<Min>

***{WatchField | SingleValue}***

</Min>

<Max>

***{WatchField | SingleValue}***

</Max>

<Value>

***{WatchField | SingleValue}***

</Value>

</WatchParameter>

</WatchRule>

### WatchParameter

Elemente Min / Max / Value sind alle optional und werden verwendet um eine Obergrenze / Untergrenze bzw. genaue Wertüberwachung für das **WatchField** zu definieren.

* Min

Clients werden notifiziert wenn sich der **WatchField** Wert diesen Min Wert entspricht bzw. **unterschreitet**.

* Max

Clients werden notifiziert wenn sich der **WatchField** Wert diesen Max Wert entspricht bzw. **überschreitet**.

* Value

Clients werden notifiziert wenn der **WatchField** Wert diesen Value Wert **entspricht**.

### Attribute

* **Name**

Gibt einen eindeutigen Name für diese Regel am Service System an.

* **Notify**

Beschreibt ob für diese Regel Notifications versendet werden sollen.

* **Period**

Beschreibt wie lange der Service zwischen den Auslesungsvorgängen des angegebenen WatchFields wartet und diesen Wert im Anschluss mit dem angegebenen WatchParametern abgleicht. Standardwert ist 60 Sekunden.

* **CacheValues**

Gibt an wie viele ausgewerte Watch Values der Service behalten soll. Der Standard Wert beträgt 50 zwischengespeicherte Werte. Diese Werte gehen beim Restart des Services verloren.

## OperationStatus

Das Element **OperationStatus** wird in Form einer Message als Returnwert aus viele Konfigurationsmessages vom Service zurückgegebenen.

<OperationStatus>

<Command> *# CommandName (Bsp: DeleteWatchRule) #* </Command>

<Status> *# OK | INVALIDINPUT | ERROR #* </Status>

<Message> *# Detailed Error Message #* </Message>  
</OperationStatus>

## Process

Das Element **Process** wird verwendet um einen Process am Hostsystem der Services darzustellen.

<Process>

<ProcessName> *# Name des Prozess #* </ProcessName>

<RAM> *# Aktuelle RAM auslastung #* </RAM>

<ProcessDescription> *# Beschreibung #* </ProcessDescription>

<PID> *# ProcessID #* </PID>

<User> *# Owner Benutzer des Prozess #* </User>

</Process>

## Setting

Das Element **Setting** wird dazu verwendet um eine Einstellung des Services abzubilden.

<Setting Key=” *# Settingskey #* ” Value=” *# Settingsvalue #* ” />

# Messages

RemRes verwendet diese Messages zur Konfiguration der Überwachung des Computers. In diesen Messages werden die definierten Grundbausteine möglichst oft wiederverwendet.

## AddWatchRule

Mit der Message **AddWatchRule** kann dem RemRes Service eine neue **WatchRule** mitgeteilt werden.

<AddWatchRule>

***{WatchRule}***

</AddWatchRule>

Als Result wird eine **OperationStatus** Message zurückgegebene.

## NotifyMe

Mit der Message **NotifiyMe** wird dem Service eine Client mitgeteilt der in Zukunft notifiziert werden soll. Die **NotifyMe** Message sollte bei jedem Start des Clients erfolgen. Da der RemRes Service den NotificationEndPunkt nach einer definierbaren Zeitspanne verwirft.

<NotifyMe>

<Endpoint> *#Endpunkt des Notification Clients#* </Endpoint>

</NotifyMe>

Als Result wird eine **OperationStatus** Message zurückgegebene.

## ClearWatchRules

Mit der Message **ClearWatchRules** wird dem Service mitgeteilt alle aktuellen WatchRule zu verwerfen.

<ClearWatchRules/>

Als Result wird eine **OperationStatus** Message zurückgegebene.

## DeleteWatchRule

Mit der Message **DeleteWatchRule** wird dem Service mitgeteilt die aktuelle **WatchRule** mit dem angegebenen Namen zu verwerfen.

<DeleteWatchRule Name=” *#Name der WatchRule#* ” />

Als Result wird eine **OperationStatus** Message zurückgegebene.

## GetWatchData

Mit der Message **GetWatchData** wird dem Service mitgeteilt das er geloggte Datenbestände an den anfordernden Client senden soll.

### All Watched Data

Diese Message fordert den Service auf **alle geloggten Daten** zu retournieren. Diese geloggten Daten kommen durch die Gesamtheit aller definierten **WatchRules** zustande.

<GetWatchData/>

### For a specified WatchField

Diese Message fordert den Service auf **geloggte Daten** für dieses **WatchField** zu retournieren.

<GetWatchData>

<WatchField>

<WatchObject> *#WMI Object#* </WatchObject>

<WatchProperty> *#WMI Object Property#* </WatchProperty>

</WatchField>

</GetWatchData>

### For a specified WatchRule

Diese Message fordert den Service auf **geloggte Daten** für die durch den angegebenen Namen identifizierte **WatchRule** und dadurch ein **WatchField** zu retournieren.

<GetWatchData Name=“ *#Watch Rule Name#* “ />

### Result

<GetWatchDataResult>

<WatchDataSet>

<WatchField>

<WatchObject> *#WMI Object#* </WatchObject>

<WatchProperty> *#WMI Object Property#* </WatchProperty>

<WatchFieldValues>

<WatchValue>

<Value> *#Ein Ausprägungswert Wert#* </Value>

<MomentOfMeasure>*#Messzeitpunkt#*</...>

</WatchValue>

...

<WatchValue>

<Value> *#Ein Ausprägungswert Wert#* </Value>

<MomentOfMeasure>*#Messzeitpunkt#*</...>

</WatchValue>

***{0 ... n}***

</WatchFieldValues>

</WatchField>

***{0 ... n}***

</WatchDataSet>

</GetWatchDataResult>

## GetWatchRules

Die Message **GetWatchRules** wird verwendet um alle aktiven **WatchRules** vom Service abzurufen.

<GetWatchRules/>

### Result

<GetWatchRuleResult>

<WatchRuleSet>

<WatchRule Name=”*#Name of Rule#*” Notify=”*#true/false#*”

Period=”*#Ueberwachungsperiode#*” >

<WatchField>

<WatchObject> *#WMI Object#* </WatchObject>

<WatchProperty Type=”(Integer|Double|String|DateTime)”>

*#WMI Object Property#*

</WatchProperty>

</WatchField>

<WatchParameter>

<Min>

***{WatchField | SingleValue}***

</Min>

<Max>

***{WatchField | SingleValue}***

</Max>

<Value>

***{WatchField | SingleValue}***

</Value>

</WatchParameter>

</WatchRule>

***{0 ... n}***

</WatchRuleSet>

</GetWatchRuleResult>

## Notification

Treten am Service verschiedene Ereignisse ein wie z.B.: das ein Min Value einer **WatchRule** unterschritten wird generiert der Service eine **Notification** Message und sendet diese an den/die Client(s).

<Notification>

<Type> *# Notificationtype (Bsp: MaxReached) #* </Type>

<Message> *# Detailed Notification Message #* </Message>

<WatchRuleName> *# WatchRule Name#* </WatchRuleName>

<LastValue> *# Last Meeasured Value #* </LastValue>

***{WatchField}***  
</Notification>

Die Elemente **WatchRuleName**, **LastValue** und **WatchField** sind **optional**!

## Keep Alive

Die Keep Alive Messages werden verwendet um den Computer Status des entfernten Computers zu überwachen. Wenn ein Client eine **KeepAliveRequest** Message sendet muss der Client ein **KeepAliveResponse** Message senden.

### Keep Alive Request

<KeepAliveRequest/>

### Keep Alive Response (Result)

<KeepAliveResponse/>

## GetProcessList

Mit der Message **GetProcessList** kann das aktuelle Process Abbild des Host Computers des Service angefordert werden.

<GetProcessList RAMOver=” *# min. RAM auslastung #* ” />

Das Attribut **RAMOver** ist optional für würde die Process List auf Processe beschränken die zum Abfragezeitpunkt über dieser Grenze liegen.

### Result

<GetProcessListResult>

<ProcessSet>

<Process>

<ProcessName> *# Name des Prozess #* </ProcessName>

<RAM> *# Aktuelle RAM auslastung #* </RAM>

<ProcessDescription> *# Beschreibung #* </...>

<PID> *# ProcessID #* </PID>

<User> *# Owner Benutzer des Prozess #* </User>

</Process>

***{0 ... n}***

</ProcessSet>

</GetProcessListResult>

## Settings

Der RemRes Service verfügt über verschiedene Einstellungswerte diese können mit dem Settings Message Set verwaltet werden.

### GetSetting

Die **GetSetting** Message wird dazu verwendet um Einstellungen des Services abzurufen.

<GetSetting Key=”*# Settngskey #*” />

Die Message kann auch ohne dem **Key** Attribut gesendet werden, was zur Folge hat das alle Settings Werte zurückgegeben werden.

#### Result

<GetSettingResult>

<Settings>

<Setting Key=” *# Settingskey #* ” Value=” *# Settingsvalue #*” />

***{0 ... n}***

</Settings>

</GetSettingResult>

### SetSetting

Die **SetSetting** Message wird dazu verwendet um Einstellungen des Service zu verändern.

<SetSetting Key=”*# Settngskey #*” Value=”*# Neues Settingsvalue #*” />

Als Result wird eine **OperationStatus** Message zurückgegeben.

# Glossar

### WMI

Windows Management Instrumentation.   
Über WMI kann lesend und schreibend, auf nahezu alle Einstellungen eines Windows-Computers zugegriffen werden.

### WQL

WMI queries use WMI Query Language (WQL), a subset of SQL.