

# DISAG JSON Schnittstelle Lightweight

V0.1	06.02.2018	CRO	Erste Fassung
V0.2	08.02.2018	CRO	OSS Einstellungen hinzugefügt
V0.3	14.02.2018	CRO	Series und Result hinzugefügt

## Inhaltsverzeichnis

Allgemein, Sinn und Zweck .....	1
OUT-JSONInterface.log .....	2
UDP Broadcast .....	2
Struktur der JSON Objekte .....	3
Message .....	3
Shot .....	3
Series .....	4
Result .....	4
Shooter .....	4
Club .....	4
Team .....	5
MenuItem .....	5
Enumerationen .....	5
MessageVerb .....	5
MessageType .....	5
DiscType .....	6
TrafficLightStatus .....	6
Einstellungen in der OSS .....	6

## Allgemein, Sinn und Zweck

Die JSON Schnittstelle liefert in Echtzeit Informationen über die aktuelle Nutzung einer OpticScore-Anlage. Sie wird aus der OpticScoreServer-Software heraus bereitgestellt. Zudem werden Endgeräte

über die Schnittstelle gesteuert, wenn eine parallele, gleichzeitige Ausführung von Befehlen auf mehreren Geräten notwendig ist.

### OUT-JSONInterface.log

Die Log/Text-Datei wird unter %ProgramData%\DisagOpticScore erzeugt kontinuierlich beschrieben. Überschreitet die Datei eine Größe von 5MB, wird die aktuelle Datei nach OUT-JSONInterface.log.1 verschoben und OUT-JSONInterface.log weiter beschrieben. Es werden maximal zwei Dateien gehalten. In die Log-Datei werden nur Messages vom Typ „Event“ geschrieben.

Beim Öffnen der Datei muss darauf geachtet werden, dass diese nicht im exklusiven Zugriff geöffnet wird, da die OSS die Datei sonst nicht befüllen kann.

### UDP Broadcast

Alle Informationen werden parallel zur Logdatei über einem UDP Broadcast auf Port 30169 im lokalen Netzwerk verteilt. Mit einem Client können die Daten auf diesem Port abgegriffen werden.

Beispiel-Client unter Java:

```

1 package de.disag.jsoninterface;
2
3 import java.net.DatagramPacket;
4 import java.net.DatagramSocket;
5
6 public class BroadcastListener {
7
8     public static void main(String[] args) {
9         int nPort = 30169;
10
11         if (args.length > 0) {
12             nPort = Integer.parseInt(args[0]);
13         }
14
15         try {
16             DatagramSocket dsocket = new DatagramSocket(nPort);
17             byte[] buffer = new byte[4096];
18
19             DatagramPacket packet = new DatagramPacket(buffer, buffer.length);
20
21             while (true) {
22                 dsocket.receive(packet);
23
24                 String msg = new String(buffer, 0, packet.getLength());
25                 System.out.println(packet.getAddress().getHostName() + ": " + msg);
26
27                 packet.setLength(buffer.length);
28             }
29         } catch (Exception e) {
30             e.printStackTrace();
31         }
32     }
33 }
34

```

## Struktur der JSON Objekte

Alle Objekte besitzen das Attribut „UUID“. Dabei handelt es sich um eine UUIDv4 (uppercase), z. B.: 6727B7C5-D772-4419-ABCC-A84098F3C91C.

### Message

Parameter	Typ	Beschreibung	Werte
<b>MessageType</b>	enum	<b>MessageType</b>	
<b>MessageVerb</b>	enum	<b>MessageVerb</b>	
<b>Ranges</b>	Command: Array<int> Event: int	Liste der Stände, die von einem Kommando betroffen sind oder Standnummer von dem ein Event ausgelöst wurde	
<b>Sequential</b>	bool	Wenn mehrere Kommandos in Objects vorhanden sind, bedeutet Sequential = true, dass diese Befehle sequentiell abgearbeitet werden müssen	
<b>Objects</b>	Array	z. B. Liste von <b>Shot</b>	

### Shot

Parameter	Typ	Beschreibung	Werte
<b>ShotDateTime</b>	DateTime	Zeitstempel des Schusses	yyyy-MM-dd HH:mm:ss.fff
<b>TLStatus</b>	enum	Zustand der Ampel während des Schusses, <b>TrafficLightStatus</b>	off, red, green
<b>LastTLChange</b>	int	Zeit in ms seit letzter Ampelschaltung	>= 0
<b>Source</b>	enum	Herkunft des Schusses; OpticScore = Messrahmen	OpticScore, RedDot
<b>Range</b>	int	Standnummer	>= 0
<b>Shooter</b>	object	<b>Shooter</b>	
<b>DiscType</b>	enum	<b>DiscType</b>	LG, LGA, LP, LPA, KK, KKA, KYFFH, ZS, ZSA, ZSTRD, LPS, LPI
<b>X</b>	int	X-Position des Schusses (vom Zentrum des Messbereichs aus)	-9000 < X < 9000
<b>Y</b>	int	Y-Position des Schusses	-9000 < Y < 9000
<b>Distance</b>	float	Teiler	0 <= Distance <= 12700
<b>Count</b>	int	Nummer des Schusses	>= 0
<b>FullValue</b>	int	Ringwert ohne Zehntel	0 <= FullValue <= 10
<b>DecValue</b>	float	Ringwert mit Zehntel	0 <= DecValue <= 10.9

<b>Run</b>	int	Durchgang innerhalb der Disziplin	$\geq 0$
<b>IsValid</b>	bool	Schuss gültig	true, false
<b>IsWarmup</b>	bool	Probeschuss	true, false
<b>IsHot</b>	bool	Wertungsschuss	true, false
<b>IsDummy</b>	bool	Generierter Schuss	true, false
<b>IsInnerten</b>	bool	Innenzehner	true, false
<b>IsShootoff</b>	bool	Stechschuss	true, false
<b>MenuItem</b>	object	<b>MenuItem</b>	
<b>Remark</b>	string	Hinweis, z. B. Ringabzug	

### Series

Parameter	Typ	Beschreibung	Werte
<b>Shooter</b>	object	Shooter	
<b>ID</b>	int	Nummer der Serie	$ID > 0$
<b>FullValue</b>	int	Serienwert ohne Zehntel	
<b>DecimalValue</b>	float	Serienwert mit Zehntel	

### Result

Parameter	Typ	Beschreibung	Werte
<b>Shooter</b>	object	Shooter	
<b>FullValue</b>	int	Ergebnis ohne Zehntel	$FullValue \geq 0 \leq (ShotCount * 10)$
<b>DecimalValue</b>	float	Ergebnis mit Zehntel	$DecValue \geq 0 \leq (ShotCount * 10.9)$

### Shooter

Parameter	Typ	Beschreibung	Werte
<b>Firstname</b>	string	Vorname	
<b>Lastname</b>	string	Nachname	
<b>Birthyear</b>	int	Geburtsjahr	$> 1900$
<b>InternalID</b>	string	Interne ID	
<b>Identification</b>	string	Passnummer	
<b>Team</b>	object	<b>Team</b>	Kann NULL sein
<b>Club</b>	object	<b>Club</b>	Kann NULL sein

### Club

Parameter	Typ	Beschreibung	Werte
-----------	-----	--------------	-------

<b>Name</b>	string	Vereinsname	
<b>ShortName</b>	string	Vereinsname (kurz)	
<b>ID</b>	string	Vereinsnummer	

#### Team

Parameter	Typ	Beschreibung	Werte
<b>Name</b>	string	Mannschaftsname	
<b>ShortName</b>	string	Mannschaftsname (kurz)	

#### Menulitem

Parameter	Typ	Beschreibung	Werte
<b>MenuID</b>	string	ID des Menüeintrags	z. B. 100_4
<b>MenuPointName</b>	string	Name des Menüpunkts	z. B. „Neue Schießzeiten“
<b>MenuItemName</b>	string	Name des Eintrags	z. B. „LG 40 Schuss“

#### Enumerationen

##### MessageVerb

Wert	Command	Event
<b>Various</b>	Wenn mehrere verschiedene Befehle in Objects vorhanden hat jeder davon hat den Parameter CommandType (entspricht <b>MessageType</b> ) für sich gesetzt.	
<b>Shot</b>		Schuss gefallen
<b>Series</b>		Serie fertig
<b>Result</b>		Gesamtergebnis fertig

##### MessageType

Wert	Beschreibung
<b>Command</b>	Der Empfänger der Message muss prüfen, ob sie für ihn von Bedeutung ist (anhand der Ranges) und ggf. ausführen
<b>Event</b>	Die Message enthält Informationen wie Schüsse, Standbelegung, etc. Das Attribut Sequential ist uninteressant. Pro Event-Message ist genau ein Objekt in Objects  Bei Events werden nur die Werte in den Objekten gesetzt, die sich geändert haben, z. B. wenn Zehntelschuss an einem Stand aktiviert wurde, werden nicht alle Einstellungen geliefert

## DiscType

Wert	Beschreibung
<b>LG</b>	Luftgewehr
<b>LGA</b>	Luftgewehr-Auflage
<b>LP</b>	Luftpistole
<b>LPA</b>	Luftpistole-Auflage
<b>KK</b>	Kleinkalibergewehr
<b>KKA</b>	Kleinkalibergewehr-Auflage
<b>ZS</b>	Zimmerstutzen
<b>ZSA</b>	Zimmerstutzen-Auflage
<b>ZSTRD</b>	Zimmerstutzen-Traditionell
<b>KYFFH</b>	Kyffhäuserscheibe
<b>LPS</b>	DE Luftpistole Schnellfeuer
<b>LPI</b>	IT Luftpistole

## TrafficLightStatus

Wert	Beschreibung
<b>off</b>	Ampel ist aus
<b>red</b>	Ampel ist rot
<b>green</b>	Ampel ist grün

## Einstellungen in der OSS

Die Einstellungen der JSON Schnittstelle können in der OSS unter Extras -> Optionen -> „JSON Live“ vorgenommen werden.

