# Werkgroep OSF uitwisselformaat - Probleemstelling: opmaak in OSF

## Scenario 1 - meest flexibele oplossing: complete HTML code toelaten

### Omschrijving:

Ieder XML element inhoud kan in CDATA velden worden ingepakt. Hierdoor kan ook niet XML conforme inhoud (bijv. HTML) in de structuur staan.

### Voorbeeld:

|  |
| --- |
| <Titel>  <![CDATA[  <ul><li>Dit is een <em>heel</em> mooie post</li></ul>  ]]>  </Titel> |

### Voordeel:

* Flexibiliteit: Alle HTML notaties zijn mogelijk en kunnen worden gebruikt – en omdat HTML gebruikt wordt om de (structurele) opmaak van websites te bepalen is er best veel mee mogelijk
* Flexibiliteit: Als de besteksoftware een complete HTML renderengine (bijv. Gecko) inzet, zijn zelfs complete lokale (CSS) opmaakspecificaties denkbaar (denk aan <em style=”font-size:1.2em;color:red;background:#CCC”>heel</em> in plaats van alleen <em>heel</em>)
* Flexibiliteit: Er kunnen ook referenties naar externe inhoud worden geplaatst zoals bij hyperlinks gebruikelijk
* Flexibiliteit: Als de besteksoftware een complete HTML renderengine (bijv. Gecko) inzet, zijn ook links naar externe inhoud (bijv. afbeeldingen) denkbaar die meteen binnen het document getoond kunnen worden (net als op een website)

### Nadeel:

* Prestatie: De prestatie kan eronder uitgaan omdat de eigenlijke (tekstuele) inhoud in al die opmaak bij elkaar gezocht moet worden en niet als één string verkrijgbaar en in mogelijk veel opmaak omschrijving verpakt is
* Overzicht/structuur: Omdat iedereen zijn eigen idealen kan volgen en de structuur/opmaak nu vrij mag bepalen, kan het overzicht van het document eronder leiden (op dit moment worden klare structuren, afstanden en kleuren in een bestek behouden wat hiermee in het geheel gewijzigd/misbruikt kan worden)
* Veiligheid: Externe links zouden in een (gemanipuleerd) bestekbestand naar schadelijke locaties kunnen verwijzen – sterker nog zouden het aanroepen van externe scripts mogelijk kunnen zijn waarmee in een HTML renderengine/interpreter van de inhoud rekening moet worden gehouden om zulke inhoud niet uit te voeren
* Inspanning: Als de besteksoftware tot nu toe geen HTML renderengine (bijv. Gecko) inzet, kan de implementatie veel werk opleveren omdat in principe alle HTML werktuigen binnen het bestekpakket toegankelijk gemaakt zouden moeten worden (buttons voor het aanmaken van lijsten, hyperlinks, afbeeldingen, tabellen, vet, cursief, etc., etc., etc…) en er mogelijk een in het geheel andere manier van schermuitgave moet plaatsvinden (waar op dit moment mogelijk bestaande UI elementen worden gebruikt moet de uitgave mogelijk in het geheel om worden gegooid en aan een HTML renderengine – bijv. Gecko – worden overgedragen… hiervoor moet ook de rest van de OSF XML structuur in voor een HTML renderengine leesbare HTML code worden omgevormd)

## Scenario 2 - minder flexibele oplossing: inline omschrijving door attributen

### Omschrijving:

De DTD wordt uitgebreid door een set attributen met per attribuut een set mogelijke invullingen die het toestaan om de opmaak van een element te beïnvloeden. Deze attributen kunnen aan ieder XML element worden toegevoegd en de interpreter van de schermuitgave moet dan ervoor zorgen dat hij weet hoe het element dan op het scherm moet worden weergegeven om aan de attribuutdefinitie te voldoen. Zouden er ook “tooltips” toe worden gestaan (denkbaar) n daarmee vrije invullingen, dan moet de attribuutinhoud van deze velden gecodeerd worden (bijv. UrlEncode) om mogelijke ongeldige XML syntax te voorkomen.

### Voorbeeld:

|  |
| --- |
| <BestekPost data-url=”https%3A%2F%2Fmijn.voorbeeld.nl%2Fbestekpost%26meer”>  <ItemHeader>  <Titel style-decoration=”underline” style-weight=”bold”>  <![CDATA[  INTEGRAAL DAKBEDEKKINGSSYSTEEM, PARKEERDAK VOLL.GEHECHT  ]]>  </Titel>  <Id style-weight=”light” style-color=”#FF0000” style-background=”#E0E0E0”>  <![CDATA[  33.33.19-h  ]]>  </Id>  </ItemHeader>  </BestekPost> |

### Voordeel:

* Prestatie: Er vindt geen mengen van inhoud en opmaak binnen een XML-element waarde plaats waardoor een applicatie niet eerst op zoek moet gaan wat inhoud en wat opmaak is wat betere prestatie waarborgt.
* Overzicht/structuur: Door een (in de DTD vooraf bepaalde en dus beperkte) selectie van attributen en invullingen kan de controle over de algemene structuur van het document worden behouden.
* Flexibiliteit: Er kan rekening worden gehouden met wensen van bestekschrijvers om dingen te kunnen beïnvloeden zoals bijv.
  + Tekstkleur
  + Achtergrondkleur
  + Tekstdikte
  + Tekstgrootte
  + Tekstdecoratie
  + Tooltips
  + Links naar externe inhoud
  + Andere mogelijke wensen die in kaar gebracht zouden moeten worden

### Nadeel:

* Flexibiliteit: De DTD bepaalt de mogelijkheden – waar deze niet in voorziet is niet mogelijk en zal uiteindelijk ook niet van de programma’s geïnterpreteerd worden (maar dit is ook juist één van de beoogde voordelen). Dit leidt eigenlijk tot de conclusie dat de vragen niet zullen verdwijnen waarom dit nou weer niet mogelijk is of waarom dat op deze plek niet kan/mag. Een eigen uitbreiding zal gewoon nooit de complexiteit van HTML kunnen bereiken – tenminste niet zonder het wiel twee keer uit te vinden en veel resources eraan te verspillen in eindeloze uitbreidingen.

## Aanvullend – reduceren van redundantie door opmaak

### Omschrijving:

Dit is meer een aanvulling en onafhankelijk of scenario 1 of 2 wordt toegepast. Element gerelateerde informatie moet natuurlijk per element worden bepaald (zoals in scenario 2 bijv. het data-url attribuut). Maar verder zou (net als bij HTML die naar CSS classes refereren) een referentie met een GUID naar een opmaaksjabloon kunnen worden gemaakt wat een verwijzing naar een opmaakgedeelte maakt (net als de specificaties) waar de opmaakregels voor elementgroepen worden gemaakt.

### Voorbeeld in verbinding met scenario 1:

|  |
| --- |
| <Werksoorten>  <Hoofdstuk>  …  <Paragraaf>  …  <BestekPost data-url=”https%3A%2F%2Fmijn.voorbeeld.nl%2Fbestekpost%26meer”>  <ItemHeader>  <Titel style-id=”1959a890-eff9-4f7e-bbed-936513dc046e”>  <![CDATA[  INTEGRAAL DAKBEDEKKINGSSYSTEEM, PARKEERDAK VOLL.GEHECHT  ]]>  </Titel>  <Id style-id=”bd754933-2045-4d08-ab5d-643046d41f12”>  <![CDATA[  33.33.19-h  ]]>  </Id>  </ItemHeader>  </BestekPost>  </Paragraaf>  </Hoofdstuk>  </Werksoorten>  <Styles>  <Style id=”1959a890-eff9-4f7e-bbed-936513dc046e”>  tekst-decoration:underline;font-weight:bold;  </Style>  <Style id=”bd754933-2045-4d08-ab5d-643046d41f12”>  font-weight:light;color:#FF0000;background:#E0E0E0;  </Style>  </Styles> |

### Voorbeeld in verbinding met scenario 2:

|  |
| --- |
| <Werksoorten>  <Hoofdstuk>  …  <Paragraaf>  …  <BestekPost data-url=”https%3A%2F%2Fmijn.voorbeeld.nl%2Fbestekpost%26meer”>  <ItemHeader>  <Titel style-id=”1959a890-eff9-4f7e-bbed-936513dc046e”>  <![CDATA[  INTEGRAAL DAKBEDEKKINGSSYSTEEM, PARKEERDAK VOLL.GEHECHT  ]]>  </Titel>  <Id style-id=”bd754933-2045-4d08-ab5d-643046d41f12”>  <![CDATA[  33.33.19-h  ]]>  </Id>  </ItemHeader>  </BestekPost>  </Paragraaf>  </Hoofdstuk>  </Werksoorten>  <Styles>  <Style id=”1959a890-eff9-4f7e-bbed-936513dc046e”>  <Decoration>underline</Decoration>  <Weight>bold</Weight>  </Style>  <Style id=”bd754933-2045-4d08-ab5d-643046d41f12”>  <Weight>light</Weight>  <Color>#FF0000</Color>  <Background>#E0E0E0</Background>  </Style>  </Styles> |

### Voordeel:

* Efficiëntie: Vergelijkbare elementen hoeven niet per element alle opmaakregels toegewezen te krijgen maar delen allemaal een opmaaksjabloon die maar op één plek hoeft te worden bijgewerkt.
* Efficiëntie: Bij grote hoeveelheden vergelijkbare elementen kan de bestandsgrootte duidelijk verminderd worden.

### Nadeel:

* Prestatie: Opmaak is niet meer direct aan een element gekoppeld en daarmee meteen verkrijgbaar maar moet in de XML structuur eerst worden opgezocht.
* Inspanning: In het bestekprogramma moet een beheer van centrale opmaakregels worden toegevoegd wat in principe meer werk maakt dan direct attributen aan een element toe te voegen.

De in dit document genoemde attributen/structuur zijn puur ter illustratie en zijn niet als uiteindelijk voorstel bedoeld. Het gaat meer om de structurele mogelijkheden van de verschillende scenario’s. Bestekprogramma schrijvers zullen ongetwijfeld nog meer nadelen/voordelen in de details zien dan de auteur van dit document (die geen kennis van de actuele implementatie binnen de bestekpakketten heeft) en mogen hun inschatting van de scenario’s graag toevoegen. Complete eigen voorstellen/variaties (bijv. uit technische redenen) zijn ook welkom.