

Arena Arbeidsflytskript Brukerdokumentasjon

DIPS AS – Versjon 18.1, 2019-01-15 |



Innhold

1. Arena Arbeidsflytskript
 - 1.1. Hurtiginnføring for Arbeidsflytskript
 - 1.2. Bruke Arbeidsflytskript
 - 1.2.1. Godkjente bruksområder
 - 1.2.2. Overordnet struktur for skript
 - 1.2.2.1. Kjede sammen ting som skal skje
 - 1.2.2.2. Hendelse
 - 1.2.2.3. Plugin
 - 1.2.3. Opprette nytt skript
 - 1.2.3.1. Gjenbruk av skriptnavn
 - 1.2.3.2. Skjermbildereferanser
 - 1.2.4. Skrive og editere skript
 - 1.2.4.1. Context
 - 1.2.4.2. Noen eksempler
 - 1.2.5. Endre skriptnavn
 - 1.2.6. Lagre skript
 - 1.2.6.1. Kansellering og ulagrede endringer i skript
 - 1.2.6.2. Skjermbildereferanser
 - 1.2.7. Versjonering av skript
 - 1.2.8. Verifisere skript
 - 1.2.8.1. Skjermbildereferanser
 - 1.2.9. Importere skript
 - 1.2.9.1. Skjermbildereferanser
 - 1.2.10. Eksportere skript
 - 1.2.10.1. Skjermbildereferanser
 - 1.2.11. Aktivere skript
 - 1.2.11.1. Skjermbildereferanser
 - 1.2.12. Deaktivere skript
 - 1.2.12.1. Skjermbildereferanser
 - 1.2.13. Arkivere skript
 - 1.2.13.1. Skjermbildereferanser
 - 1.2.14. Gjenopprette skript
 - 1.2.14.1. Skjermbildereferanser
 - 1.3. Oppsett, konfigurasjon og tilgangskontroll
 - 1.3.1. Forutsetninger for bruk
 - 1.3.2. Tilgangskontroll
 - 1.3.2.1. Elementtyper

No part of this publication may be reproduced, stored in a retrieval system, transmitted, or published to a third party, in any form or by any means, mechanical, electronic, photocopying, recording, or otherwise, without prior written permission of DIPS AS.

OpenEHR is a registered trademark of OpenEHR Foundation. HL7®, CDA®, FHIR® and the FHIR [FLAME DESIGN]® are the registered trademarks of Health Level Seven International.

All other trademarks mentioned herein are the property of their respective owners.

DIPS AS

Postboks 1435

8037 Bodø

Norway

<https://www.dips.no>

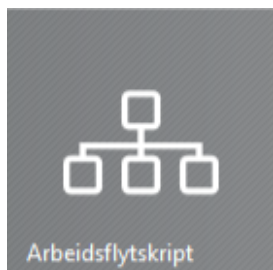
+47 75 59 20 00

Historikk

Dato	Revisjon	Forfatter	Beskrivelse
2019-01-15	1.0	Tonje Fossåskaret	Lansering av Arena 18.1

1. Arena Arbeidsflytskript

«Arbeidsflytskript» en samling instruksjoner som kan definere spesifikk oppførsel for definerte hendelser/dokumenter.



Figur 1. Flis Arbeidsflytskript

1.1. Hurtiginnføring for Arbeidsflytskript

Oppgave	Hvordan?
Legge til skript	Gjøres ved å skrive nytt skriptnavn i felt for <i>Filtrer/ Lag ny</i> og trykk på <i>Lag nytt skript</i> -knappen. Se Lage nytt skript
Endre skriptnavn	Gjøres ved å høyreklikke på skriptet og velge Endre skriptnavn. Se Endre skriptnavn
Lagre skript	Gjøres ved å trykke på <i>Lagre</i> -knappen for det aktuelle skriptet. Alternativt kan  +  benyttes. Se Lagre skript
Se alle versjoner av skript	Gjøres ved å trykke på pil-symbol ved skriptnavn. Liste over alle skriptversjoner kan ekspanderes og minimeres. Se Versjonering
Verifisere skript	Gjøres ved å trykke på <i>Verifiser</i> -knappen for det aktuelle skriptet eller trykke på <i>Verifiser alle skript</i> -knappen. Alternativt kan  +  benyttes. Se Verifisere skript
Importere skript	Gjøres ved å trykke på <i>Importer</i> -knappen og deretter velge skript som skal importeres. Se Importere skript
Eksportere skript	Gjøres ved å klikke på <i>Eksporter</i> -knappen for det aktuelle skriptet eller trykke på <i>Eksporter alle skript</i> -knappen og deretter velge lokasjon for det/ de eksporterte skriptene. Se Eksportere skript
Aktivere skript	Gjøres ved å trykke på symbol for <i>Aktiver</i> . Skriptet aktiveres og funksjoner/ hendelser definert i skriptet utføres i systemet. Se Aktivere skript
Deaktivere skript	Gjøres ved å trykke på symbol for <i>Deaktiver</i> . Skriptet deaktiveres og funksjoner/ hendelser definert i skriptet utføres ikke i systemet. Se Deaktivere skript
Arkivere skript	Gjøres ved å trykke på symbol for <i>Arkiver skript</i> , og deretter bekrefte arkivering. Se Arkivere skript
Gjennopprette skript	Gjøres ved å trykke på symbol for <i>Gjenopprett</i> Se Gjenopprette skript

Tabell 1. Hurtiginnføring

1.2. Bruke Arbeidsflytskript

1.2.1. Godkjente bruksområder

«Arbeidsflytskript» skal brukes til å redigere, vedlikeholde og utvikle nye skript som definerer en arbeidsflyt. Den gir mulighet til å opprette, importere, eksportere og endre skript.

1.2.2. Overordnet struktur for skript

Et skript inneholder en samling av regler/ et regelsett med minimum én regel. Hver regel har:

- en betingelse (true/false)
- for hvert utfall av betingelsen
 - et innhold (en sekvens av ting som må utføres)
 - en peker til neste regel

På denne måten kan du kjede sammen regler, og du bygge opp komplekse betingelsesstrukturer, samtidig som hver enkel regel kan holdes enkel.

Eksempel:

```
##=====
#RuleName = CheckIsApproved
#Condition = context["DocumentIsApproved"]
#WhenConditionDo = Plugins.Execute("CreateTaskForDocument", context["DocumentId"],
CommandTaskType.Completion, 123456)
#WhenConditionNext =
#OtherwiseDo =
#OtherwiseNext =
```

I dette eksempelet er det kun én regel, med veldig enkelt innhold.

Skript indikator	Forklaring
##	Markerer begynnelsen av reglen, for å skille de den fra andre regler i det samme skript.
Likhetstegnene	Har ingen funksjonell betydning, men er med for å gjøre skriptet mer lesbart.
#RuleName	Regelnavnet er nøkkel til å kjede sammen flere regler. Første regel i skriptet kjøres alltid først, deretter følges kjeden av "Next".
#Condition	Betingelse som bestemmer hvilken del av skriptet som skal eksekveres.
#WhenConditionDo	Hvilke ting som gjøres når betingelsen er oppfylt.
#WhenConditionNext	Navnet på neste regel som skal kjøres etter at <i>WhenConditionDo</i> er utført. Hvis ingen verdi er angitt stopper skriptet her.
#OtherwiseDo	Hvis betingelse <i>Condition</i> ikke oppfylles kjøres denne del av skriptet.
#OtherwiseNext	Regelnavnet til neste regel som skal kjøres hvis <i>Condition</i> ikke slår til.

Tabell 2. Skript forklaring

Følgende eksempel viser et skript med flere regler:

```
## =====
#RuleName = CheckWasApproved
#Condition = context["DocumentWasApproved"]
#WhenConditionDo =
#WhenConditionNext = CheckIsXrayReferral
#OtherwiseDo =
#OtherwiseNext =
## =====
#RuleName = CheckIsXrayReferral
#Condition = 2227969.Equals(context["DocumentTypeId"])
#WhenConditionDo = Plugins.Execute("CreateTaskForDocument", context["DocumentId"],
TaskTypeId.DocumentPrinting, 1001801)
#WhenConditionNext =
#OtherwiseDo =
#OtherwiseNext = CheckIsLabReply
## =====
#RuleName = CheckIsLabReply
#Condition = 1224660.Equals(context["DocumentTypeId"])
#WhenConditionDo = Plugins.Execute("CreateTaskForDocument", context["DocumentId"],
TaskTypeId.DocumentPrinting, 1001801)
#WhenConditionNext =
#OtherwiseDo =
#OtherwiseNext =
```

Første regel «CheckWasApproved» sjekker om dokumentet er i en godkjenningssprosses. Hvis dette stemmer, skal neste regel «CheckIsXrayReferral» kjøres.

«CheckIsXrayReferral» sjekker om dette er en spesifikk dokument type. I så fall gjøres et kall til en plugin (Se [Plugin](#)) med navn «CreateTaskForDocument». Denne lager en arbeidsoppgave tilknyttet dokumentet, i dette tilfellet en utskriftsoppgave, til arbeidsgruppe id 1001801.

De to id'ene, dokumenttypeid og arbeidsgruppeid, er eksempler på informasjon må finnes i den lokale systemkonfigurasjonen.

Den første regelen er da bare brukt for å sjekke basisbetingelse («WasApproved») og videre kjøring skjer i neste regel. I dette eksempelet er det altså to betingelser. I et enkelt eksempel som dette kunne de vært kombinert og hele eksempelet løst med én regel. Eksempelet viser imidlertid hvordan mer kompliserte betingelser kan brytes opp med hver sine ting som skal skje videre. Dermed blir både hver enkelt betingelse og det som skal skje i skriptet ganske enkle.

1.2.2.1. Kjede sammen ting som skal skje

Som beskrevet over er nøkkelordene «WhenConditionNext» og «OtherwiseNext» brukt til å kjede sammen flere regler. Regler kjedes sammen inntil en tom «Next», som vil avslutte sekvensen. Men hvor er begynnelsen på en sekvens? Det er alltid det første skriptet. Så det vil starte fra toppen og derfra styre sekvenser av betingelsene, fram til en tom «Next».

1.2.2.2. Hendelse

En hendelse er en "melding" fra et sted i Arena om at "noe signifikant" har skjedd. Hva som er signifikant bestemmes i de enkelte program-modulene. Hver hendelse er av en unik type som bare kan oppstå på et bestemt sted i en modul. Modulene må selv definere hendelsen:

- unik EventType på formen "workflowevent.modul.spesifikasjon", f.eks. "workflowevent.healthrecord.document.aftersave"
- en liste av verdier, navn og type, som følger eventen

Hver hendelse er selvdokumenterende og du finner en oversikt over mulige hendelser i Arena i fanen "Hendelser" i arbeidsflytskript flisen.

Skript **Hendelser** **Plugins**

◀ workflowevent.healthrecord.document.aftersave

Beskrivelse

This event occurs just after a document has been stored, whether it is a new document, a new version or an update. At this point the document and all related objects have an Id.

Påkrevde stikkord

PatientId	Int64
Document	DIPS.HealthRecord.Document
DocumentId	Int64
DocumentTypeId	Int64
DocumentWasApproved	Boolean

▶ workflowevent.healthrecord.document.beforesave

▶ workflowevent.sampleArchetypeld.aftersave

▶ workflowevent.sampleArchetypeld.beforesave

Figur 2. Liste med hendelser



Siden hendelsene er selvdokumenterende, kommer dokumentasjonen fra programkoden på serveren. Den foreligger derfor foreløpig bare på engelsk.

1.2.2.3. Plugin

Formålet med skript er å fange opp hendelser og bruke innholdet i dem til å skape lokalt tilpasset funksjonalitet. Funksjonaliteten som skapes må imidlertid reflekteres tilbake til Arena i form av arbeidsoppgaver, dokumenter eller andre "artefakter" som representerer den spesifikke funksjonaliteten. Et typisk eksempel på en hendelse er et dokument blir lagret. Dynamiske aspekter ved et dokument er dets metadata (forfatter, dokument type, størrelse) og innhold (arketyper).

Et skript kan bare eksekvere funksjonalitet som allerede finnes et eller annet sted i systemet. For å gjøre slik funksjonalitet tilgjengelig for skript på en rimelig enkel og sikker måte brukes "plugins". En plugin er en navngitt funksjon som krever bestemte parameter og som utfører sin funksjon vha. de gitte parametre og eventuelt returnere en verdi. En typisk plugin vil være en "innpakning" av en eksisterende tjeneste for å gjøre den lett å bruke i skript.

Alle skript har property "plugins", som gir tilgang til alle tilgjengelige plugins. Fra skript kalles plugin slik (se også eksemplene):

```
Plugins.Execute("PluginId", parameter1, parameter2, ...)  
var startTime = Plugins.Execute<DateTime>("PluginId", parameter1, parameter2, ...)  
Plugins.TryExecute("PluginId", parameter1, parameter2, ...)
```



Når returverdien fra en plugin skal brukes kan returtypen angis som vist. Den må selvsagt matche typen som faktisk returneres, men dette sørger for at den blir tilgjengelig på riktig måte. Uten denne angivelsen blir det bare returnert "object".



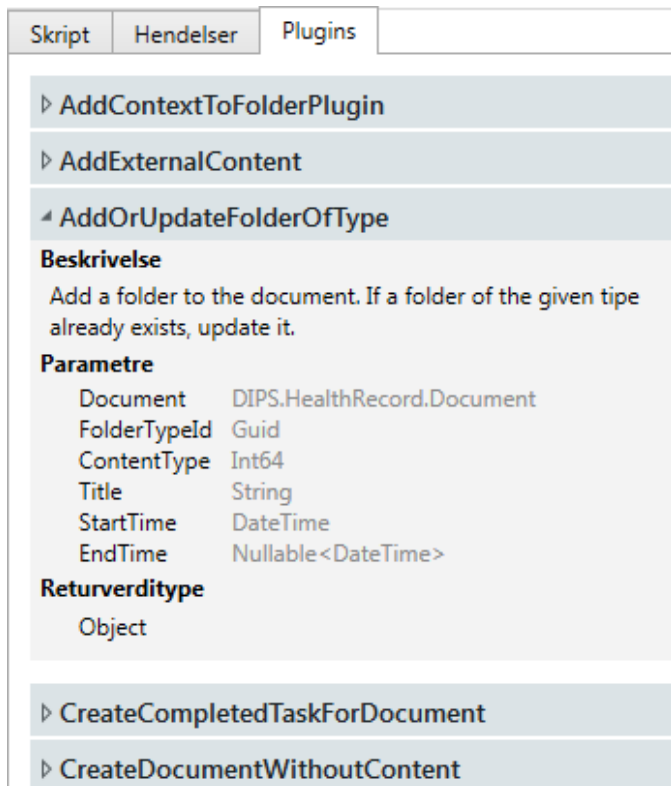
Bare typer som er kjent for skript kan brukes som type for returverdien. Dette er i praksis grunnleggende systemtyper (string, int, DateTime etc), samt noen sentrale DIPS-typer som Document og typer fra OpenEhr's referansemodell ("arketyper"). Alle andre typer må håndteres som object og kan da i praksis bare brukes som input i en annen plugin som skjønner typen.



Skriptet vil feile dersom det kaller en plugin som ikke finnes. I noen tilfeller kan det være behov å kalle en plugin, uten å feile om den ikke finnes. Til det brukes **TryExecute**.

1.2.2.3.1. Tilgjengelige Plugin

Hver plugin er selvdokumenterende og i fanen "Plugins" finner du oversikt over tilgjengelige plugins som kan brukes i skript.



Figur 3. Liste med plugins



Siden plugin er selvdokumenterende, kommer dokumentasjonen fra programkoden på serveren. Den foreligger derfor foreløpig bare på engelsk.

1.2.3. Opprette nytt skript

Skriptets navn må være unikt og bør identifisere skriptets funksjon. Utover det har selve navnet ingen betydning.

I tillegg har hvert skript to attributter, "Konsept" og "Hendelse". "Konsept" kan brukes til å gi flere skript et samlenavn, f.eks. "Pakkeforløp kreft". Utover å være en ekstra, visuell identifikator har den ingen funksjon.

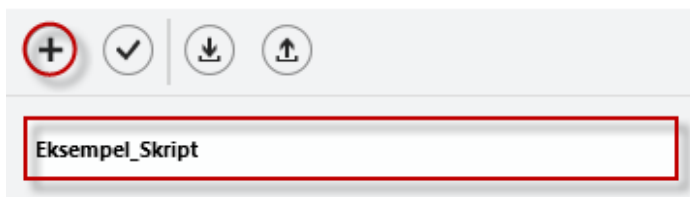
"Hendelse" derimot, er en svært viktig attributt. Denne må være identisk med en hendelse som oppstår i systemet og binder skriptet til denne hendelsen.



I tidligere versjon var hendelse lik skriptnavn, altså var skriptnavnet viktig for riktig funksjon. Dette er altså ikke tilfelle lenger. Det er likevel slik at dersom hendelse ikke oppgis så brukes skriptnavnet som hendelse. Dette er en overgangsordning og det anbefales å alltid oppgi hendelse eksplisitt.

Nytt skript opprettes ved å:

1. Skriv inn navnet til skriptet i filterboksen.
2. Klikk på «Legg til nytt skript»-knappen eller skriv skriptnavn i felt og trykk "Enter-knapp".
3. Skriv inn konsept og hendelse.
4. Skriv skriptets innhold i skripteditoren.
5. Lagre skriptet.



Figur 4. Lage nytt skript

1.2.3.1. Gjenbruk av skriptnavn

Dersom du ved opprettelse av nytt skript gjenbruker et skriptnavn som allerede finnes i databasen, vil skripteditoren ved lagring av skriptnavnet hente siste versjon av skriptet som presenteres i skripteditoren.

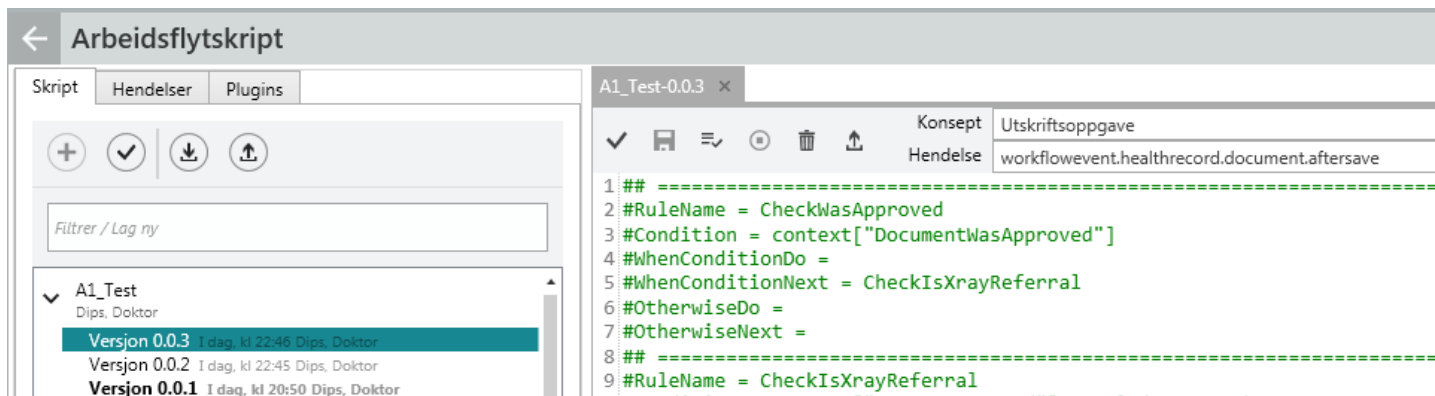
1.2.3.2. Skjermbildereferanser

Skjermelement	Beskrivelse
Lag nytt skript	Knappen oppretter skript med navnet du har angitt i felt for skriptnavn. Skriptnavn er ikke case-sensitiv.

Tabell 3. Skjermfeltbeskrivelser

1.2.4. Skrive og editere skript

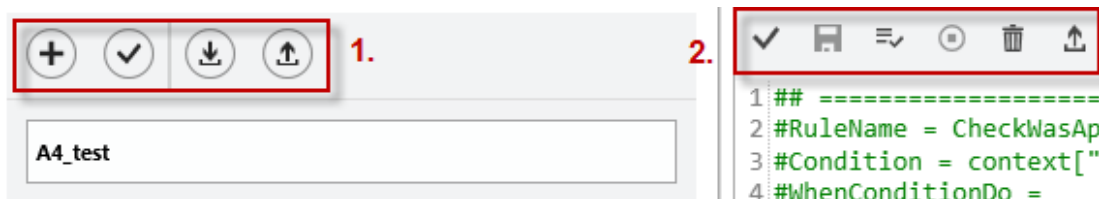
Skript skrives og redigeres i editoren i Arena:



Figur 5. Arbeidsflytskript i Arena

Ved editering av skript i flisen er det to knapperader:

1. Knappene som gjelder alle skript ligger over skriptlisten.
2. Knapper som gjelder det enkelte skript ligger under fanen i selve editoren.



Figur 6. Menylinje Arbeidsflytskript

Alle versjoner av skript vises, men det er kun nyeste versjon som kan editeres i. Se [Versjonering](#) for nærmere informasjon.

Skript kan eksporteres og importeres, og kan derfor også lages eller endres med en tekst editor.



Eldre versjoner av skript kan ikke editeres i skripteditoren i flisen, men innhold fra en eldre versjon kan kopieres til bruk i en nyere versjon av skriptet.

1.2.4.1. Context

Når et skript blir kjørt, blir et parameter kalt «context» tilgjengelig. «context» er i utgangspunktet listen over verdier som følger hendelsen. Skriptet kan imidlertid legge inn sine egne verdier og disse blir da tilgjengelige for resten av skriptet.



«Context» kan ha redundante verdier. F.eks. kan hendelser på dokument inneholde dokumentet og i tillegg ha parametere som lett kunne vært hentet ut fra dokumentet. Dette er for å forenkle skriptene for de mest typiske tilfellene.



«Context» defineres i hvert tilfelle av den hendelsen som behandles. En gitt hendelsestype har alltid samme context.

1.2.4.2. Noen eksempler

Under er det beskrevet og illustrert bruk av hhv. hendelse og plugin i skript.

Hendelse i skript

Bildet under illustrerer bruk av hendelse i skript. Hendelsen "workflowevent.healthrecord.document.beforesave" er brukt i skriptet med tilhørende verdier som følger hendelsen. Samtidig viser dokumentasjonen og illustrasjonen sammenhengen mellom de ulike delene i dokumentasjonen og i bruken av hendelse og context i skriptet.

Arbeidsflytskript

Skript | **Hendelser** | **Plugins**

▶ workflowevent.healthrecord.document.aftersave

◀ workflowevent.healthrecord.document.beforesave

Beskrivelse

This event occurs just before a document is being stored, whether it is a new document, a new version or an update. At this point the document and related objects may not have an Id.

Påkrevde stikkord

PatientId	Int64
Document	DIPS.HealthRecord.Document
2. DocumentId	Int64
DocumentTypeId	Int64
1. DocumentWasApproved	Boolean

▶ workflowevent.sampleArchetypeId.aftersave

▶ workflowevent.sampleArchetypeId.beforesave

workflowevent.openEHR-EHR-INSTRUCTION.trajectory.v1.beforesave

Konsept: Pakkeforløp Kreft

Hendelse: workflowevent.openEHR-EHR-INSTRUCTION.trajectory.v1.beforesave

```

1 ## =====
2 #RuleName = CancerTrajectoryInstructionApproved
3 #Condition = context["DocumentWasApproved"]
4 #WhenConditionDo = 1.
5 const long ArenaFolderSummaryFormSystemId = 268844;
6 context["local_scriptArchetype"] = "openEHR-EHR-INSTRUCTION.trajectory.v1";
7 var trajectoryTypePath = string.Format("/content[openEHR-EHR-SECTION.adhoc.v1]/items[{0}]/activities[at
8 Plugins.Execute("AddExternalContent", context["Document"], ArenaFolderSummaryFormSystemId, "CancerTraje
9 context["local_TrajectoryType"] = Plugins.Execute<DvText>("RMGetDvText", context["Document"], trajectory
10 #WhenConditionNext = CheckTrajectoryTypeNull 2.
11 #OtherwiseDo =
12 #OtherwiseNext =
13
14 ## =====
15 #RuleName = CheckTrajectoryTypeNull
16 #Condition = context["local_TrajectoryType"] == null
17 #WhenConditionDo =
18 ReportError(string.Format("Workflowskript {0}: Error retrieving trajectory type from document.", contex
19 #WhenConditionNext =
20 #OtherwiseDo =
21 var startTimePath = string.Format("/content[openEHR-EHR-SECTION.adhoc.v1]/items[{0}]/activities[at0001]
22 context["local_starttime"] = Plugins.Execute<DateTime?>("RMGetDvDateTimeValue", context["Document"], st
23 #OtherwiseNext = CheckStartTimeNull
24

```

Figur 7. Eksempel på bruk av hendelse og context i skript

Plugin i skript

Bildet under viser både dokumentasjon og illustrerer bruk av plugin i skript. Plugin "GetFolderIdOfType" er brukt i skriptet med tilhørende parametre og returverditype. Illustrasjonen viser sammenhengen mellom de ulike delene i dokumentasjonen og i bruken av plugin i skriptet, herunder bruk av parametre fra context.

The screenshot shows the Arena Workflowscript editor. On the left, there is a sidebar with a tree view of plugins. The 'GetFolderIdOfType' plugin is selected, and its documentation is displayed. The documentation includes a description, parameters (Document, DIPS.HealthRecord.Document, FolderTypeId, Guid), and return types (Guid). The main area shows a script snippet that uses the 'GetFolderIdOfType' plugin. The script is written in a C#-like syntax. The script snippet is as follows:

```
1 ## =====
2 #RuleName = CancerTrajectoryActionApproved
3 #Condition = context["DocumentWasApproved"]
4 #WhenConditionDo =
5 const string followupActionPath = "/content[openEHR-EHR-SECTION.adhoc.v1]/items[openEHR-EHR-ACTION.trajectory_follow_up.v1]";
6 context["local_followupaction"] = Plugins.Execute<Entries.Action>("RMGetAction", context["Document"], followupActionPath);
7 #WhenConditionNext = CheckHasAction
8 #OtherwiseDo =
9 #OtherwiseNext =
10 ## =====
11 #RuleName = CheckHasAction
12 #Condition = (Entries.Action)context["local_followupaction"] != null
13 #WhenConditionDo =
14 var action = (Entries.Action)context["local_followupaction"];
15 context["local_endTime"] = action.Time.AsDateTime;
16 context["local_state"] = action.IsmTransition.CurrentState.DefiningCode.CodeString;
17 #WhenConditionNext = CheckTrajectoryEnded
18 #OtherwiseDo =
19 #OtherwiseNext =
20 ## =====
21 #RuleName = CheckTrajectoryEnded
22 #Condition = new [] { "528", "531", "532", "533" }.Contains((string)context["local_state"])
23 #WhenConditionDo =
24 var CancerTrajectory = new Guid("{e96de319-e2a2-cc6c-e053-146a000abdb9}");
25 context["local_trajectoryfolder"] = Plugins.Execute<Guid>("GetFolderIdOfType", context["Document"], CancerTrajectory);
26 #WhenConditionNext = UpdateFolders
27 #OtherwiseDo =
28 #OtherwiseNext =
29 ## =====
30 #RuleName = UpdateFolders
31 #Condition = Guid.Empty.Equals(context["local_trajectoryfolder"])
32 #WhenConditionDo =
33 ReportError("Hendelsesregistrering kan kun gjøres fra pasientliste (1)");
34 #WhenConditionNext =
35 #OtherwiseDo =
36 Plugins.Execute("SetFolderEndTime", context["Document"], context["local_trajectoryfolder"], context["local_endTime"]);
37 #OtherwiseNext =
38 ## =====
```

The script snippet is annotated with red circles and numbers 1 through 4, corresponding to the parameters and return types in the plugin documentation. The annotations are as follows:

- 1. context["Document"]
- 2. CancerTrajectory
- 3. context["local_trajectoryfolder"]
- 4. Guid

Figur 8. Eksempel på bruk av plugin i skript

Eksempel

Reglenes betingelse (#Condition) må være et bools uttrykk (true/false). Det følgende er eksempler for en tenkt hendelse:

- context["Document_WasApproved"]
- context["Document_IsNew"] && context["Document_DocumentTypeId"] == 123456789

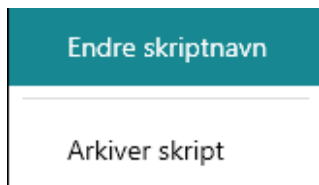
Verdier som på denne måten ekstraheres fra «context» kan brukes både som betingelse og for videre prosessering. I det følgende eksemplet hentes en verdi fra en arketype og lagres i en lokal variabel. Deretter brukes denne variabelen som parameter i en plugin. Det vises også hvordan verdier fra context kan brukes direkte som parameter til plugin:

```
var someFolderType = Guid.Parse("796b63dd-18ec-e767-e043-146a000a236d");
var trajectoryType = Plugins.Execute<DvText>("RmGetDvText ("/content[openEHR-EHR-INSTRUCTION.trajectory.v1]/activities[at0001]/description[at0002]/items[at0003]");

Plugins.Execute("CreateFolder",
    trajectoryType.Value,
    someFolderType,
    context["DocumentId"]);
```

1.2.5. Endre skriptnavn

Skriptets navn kan endres ved å høyreklikke på skriptet og trykke «Endre skriptnavn».




Figur 9. Endre skriptnavn

1.2.6. Lagre skript

Når du har endringer i et skript, vil dette vises med en grå stolpe til venstre for skriptnavnet og en stjerne til høyre. «Lagre»-knappen blir gjort tilgjengelig og det er mulig å lagre skriptet til databasen. Det er også mulig å bruke hurtigtast **Ctrl** + **S**.

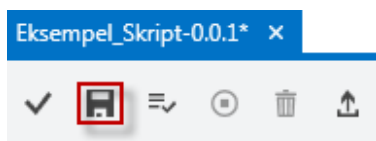
Hver gang du lagrer skriptet vil det opprettes en ny versjon, slik at du til enhver tid kan se alle lagrede endringer som er gjort for et skript. Se [Versjonering](#) for informasjon.



Et skript som er lagret kjøres ikke i systemet før etter det er godkjent og aktivert. Fom. versjon 17.2 av Arena kan du lage nye skriptversjoner som du editerer og lagrer samtidig som en tidligere versjon av skriptet er aktivert og kjører i systemet.



Figur 10. Visualisering av endringer i skriptet



Figur 11. «Lagre»-knapp er aktiv etter endring i skriptet

1.2.6.1. Kansellering og ulagrede endringer i skript

Ved kansellering av endringer i skript har du mulighet til å bruke hurtigtast **Ctrl** + **Z**. Alternativt kan du lukke skriptet uten å lagre endringer. Ved åpning av skriptet pånytt vil skriptets siste lagrede tilstand vises.

Dersom du har ulagrede endringer i skript når du logger av eller avslutter Arena, vil du få informasjon om dette slik at du har mulighet til å lagre endringene.

Du får også informasjon om ulagrede endringer når du lukker skripteditoren til et skript med ulagrede endringer.

1.2.6.2. Skjermbildereferanser

Skjermelement	Beskrivelse
Grå stolpe til venstre for skriptnavn, og stjerne til høyre for skriptnavnet	Grå stolpe og stjerne indikerer at skriptet har ulagrede endringer
Lagre	Knappen for lagring av det aktuelle skriptet du har åpnet i skripteditoren.

Tabell 4. Skjermfeltbeskrivelser

1.2.7. Versjonering av skript

Hver gang du lager skriptet blir det opprettet en ny versjon. Alle lagrede endringer som er gjort for skriptet er derigjennom tilgjengelige, og det vises hvem som har lagret endringer og tidspunkt.

Tilgjengelige versjoner av et skript er vist i liste under skriptnavnet, og det er mulig å minimere og ekspandere visning av skriptversjoner i skriptlisten. Når du åpner en versjon av et skript i editoren er versjonsnummeret også vist ved siden av skriptnavnet.

Versjoner og endring skript

Ved endring av skript er det kun nyeste versjon du kan editere, men du kan kopiere innhold fra tidligere versjoner til nyeste versjon.

Versjoner og aktivering skript

Alle versjoner av skript som kompilerer kan aktiveres, men kun en versjon kan være aktiv om gangen for et skript.

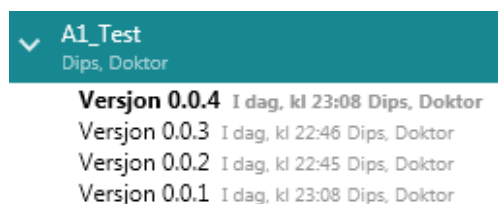
Versjoner og arkivert skript

Alle versjoner kan sees også når skriptet er arkivert.

Versjoner og eksport av skript

Hver enkelt versjon av skriptet kan eksporteres. Ved eksport angis ikke versjonsnummer, kun skriptnavn. Ved eksport av flere versjoner, vær oppmerksom på at disse vil overskrive hverandre om de lagres på samme lokasjon.

Skriptversjoner

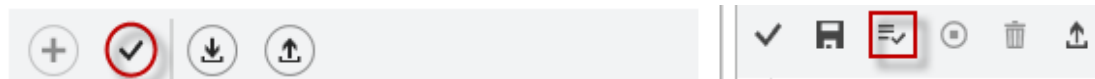


Figur 12. Flere versjoner av skript. Nyeste versjon 0.0.4 er aktivt.

1.2.8. Verifisere skript

Ved å verifisere skriptene vil du kunne se om skriptene kompilerer og er klare til bruk. Dersom et skript ikke kompilerer, vil dette vises med et rundt symbol med utropstegn. Det er mulig å se hva som har forårsaket problemene ved å klikke på [«Error List»](#) som åpner et vindu med en liste over kompileringsfeilene. Kompileringsfeilene som blir vist i listen kommer fra .NET sin C# kompilator. Det er også mulig å bruke hurtigtast **Ctrl + G**.

«Verifiser skript»-knappen



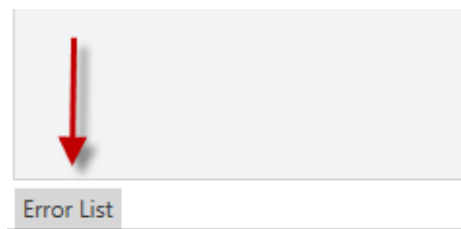
Figur 13. «Verifiser skript»-knappen

Visualisering av skript som ikke kompilerer



Figur 14. Visualisering av skript som ikke kompilerer

«Error List»



Figur 15. Knapp for «Error List»

Liste over kompileringsfeil

Error List
The name 'eFacts' does not exist in the current context
Only assignment, call, increment, decrement, await, and new object expressions can be used as a statement
The name 'yoloForDips' does not exist in the current context
Error List

Figur 16. Liste over kompileringsfeil

1.2.8.1. Skjermbildereferanser

Skjermelement	Beskrivelse
Verifiser alle skript	Knappen verifiserer alle skript
Verifiser	Knappen for verifisering av det aktuelle skriptet du har åpnet i skripteditoren.
Rødt varselstegn til venstre for skriptnavn og stjerne til høyre for skriptnavnet	Rødt varselstegn og stjerne indikerer at skriptet ikke kompilerer
Errorlist	Knappen åpner liste med kompileringsfeil

Tabell 5. Skjermfeltbeskrivelser

1.2.9. Importere skript

For å importere ett eller flere skript klikker du på «Importer skript»-knappen. Et vindu åpnes og du kan velge ett eller flere skript. Dersom du importerer et skript som allerede finnes må du velge om du vil skrive over eksisterende skript, lage en kopi eller la være å importere det.

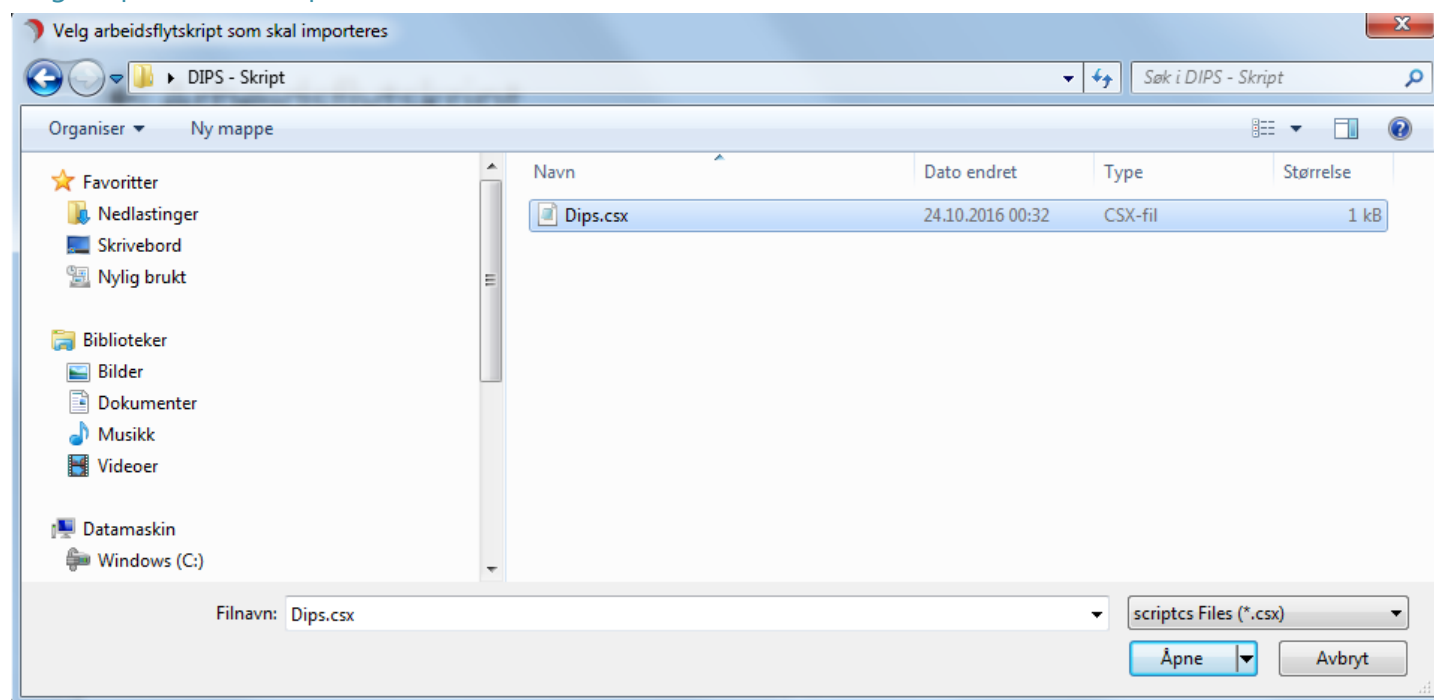
Ved import av et allerede eksisterende skript, blir nysete versjon av skript (enten det er aktivt eller ikke) overskrevet om du velger overskrive eksisterende skript i importen. Når du lagrer etter import opprettes ny versjon av skriptet som du kan aktivere.

«Importer skript»-knappen



Figur 17. Knapp for importer skript

Velg skript som skal importeres



Figur 18. Velge skript som skal importeres

Importer eksisterende skript

Importer skript

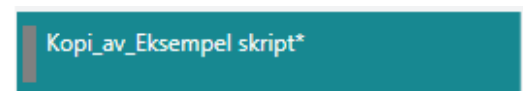
Disse skriptene finnes fra før. Kryss av hva som skal gjøres med dem:

SKRIPTNAVN	SKRIV OVER	LAG KOPI	HOPP OVER
Dips	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>

Importer
Avbryt

Figur 19. Importere eksisterende skript

Kopi av skript



Figur 20. Kopi av skript

1.2.9.1. Skjermbildereferanser

Skjermelement	Beskrivelse
Importer skript	Knappen for import av skript. Vindu for angivelse av hvor du ønsker skript importert fra vises.

Tabell 6. Skjermfeltbeskrivelser

1.2.10. Eksportere skript

For å eksportere skript klikker du på «Eksporter skript»-knappen. Et vindu åpnes og du får mulighet til å velge lokasjon på PC-en din hvor skriptene blir lagret.

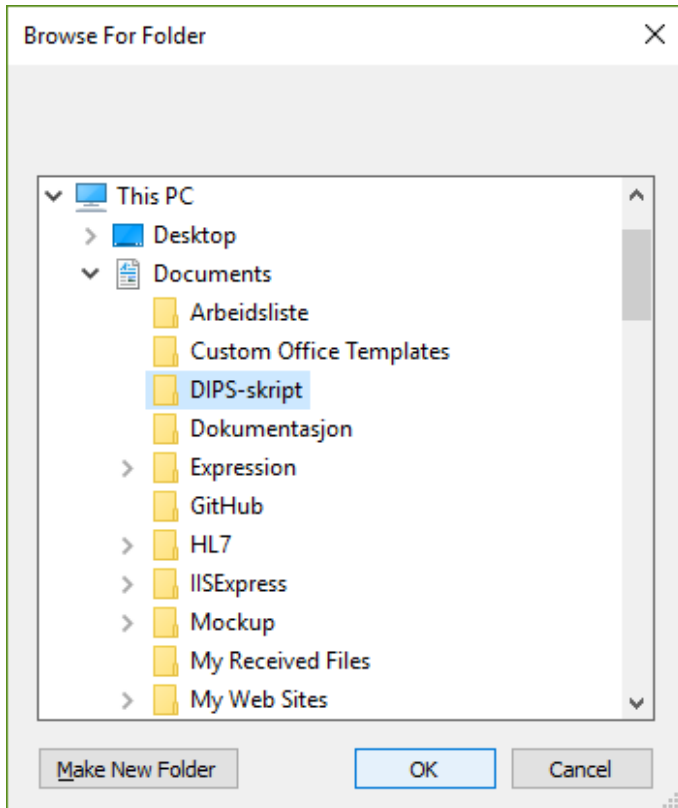
Ved eksport av skript lagres eksportert skript med skriptnavnet og innhold for gjeldende versjon som er valgt eksportert. Vær oppmerksom på at ved eksport av flere versjoner av samme skript, vil skriptinnhold overskrives dersom samme lokasjon for eksport brukes for alle versjonen.

«Eksporter skript»-knappen



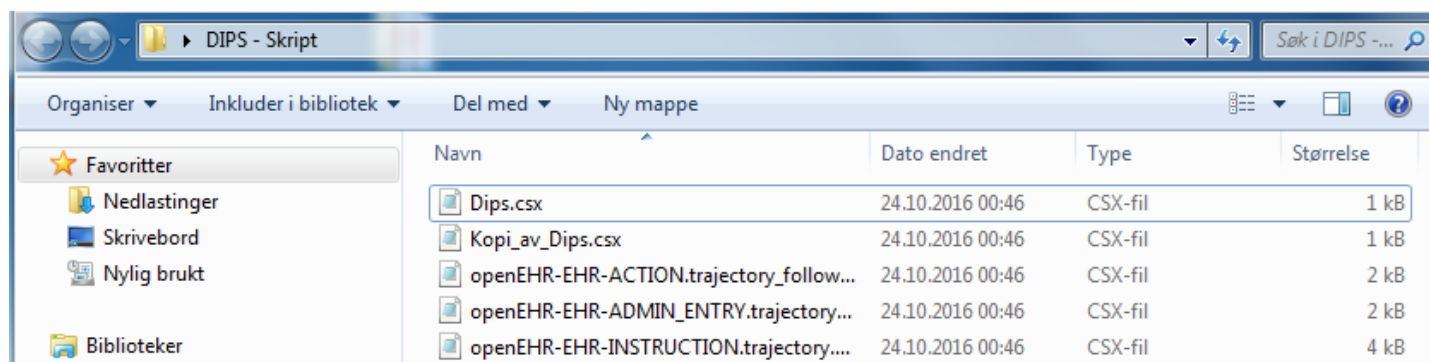
Figur 21. «Eksportere skript»-knappen

Vindu for å velge lokasjon



Figur 22. Velge lokasjon skript skal eksporteres til

Eksporterte skript



Figur 23. Eksportere skript

1.2.10.1. Skjermbildereferanser

Skjermelement	Beskrivelse
Eksporter alle skript	Knappen for eksportering av alle aktive skript i skriptlisten. Vindu for angivelse av hvor du ønsker skriptene eksportert til vises.
Eksporter	Knappen for eksportering av det aktuelle skriptet du har åpnet i skripteditoren.

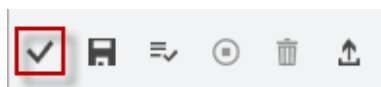
Tabell 7. Skjermfeltbeskrivelser

1.2.11. Aktivere skript

Skript som ønskes kjørt i systemet må aktiveres. For å aktivere et skript trykker du på valgt versjon av skriptet du ønsker skal aktiveres, og deretter på knappen for "Aktiver".

Ved flere versjoner av skript er det kun en versjon som kan være aktiv om gangen.

«Aktivere skript»-knappen



Figur 24. Aktivere skript

1.2.11.1. Skjermbildereferanser

Skjermelement	Beskrivelse
Godkjenn og aktiver	Knappen for "Aktiver" av det aktuelle skriptet du har åpnet i skripteditoren.

Tabell 8. Skjermfeltbeskrivelser

1.2.12. Deaktivere skript

Det er mulig å sette en versjon av et skript i deaktivert tilstand, det vil si at skriptet og skriptversjonene ligger i listen over aktuelle skript, men kjøres ikke.

«Deaktivere skript»-knappen



Figur 25. Deaktivere skript

1.2.12.1. Skjermbildereferanser

Skjermelement	Beskrivelse
Aktiver	Knappen for "Deaktiver" av det aktuelle skriptet du har åpnet i skripteditoren.

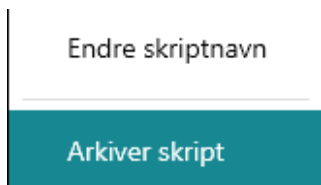
Tabell 9. Skjermfeltbeskrivelser

1.2.13. Arkivere skript

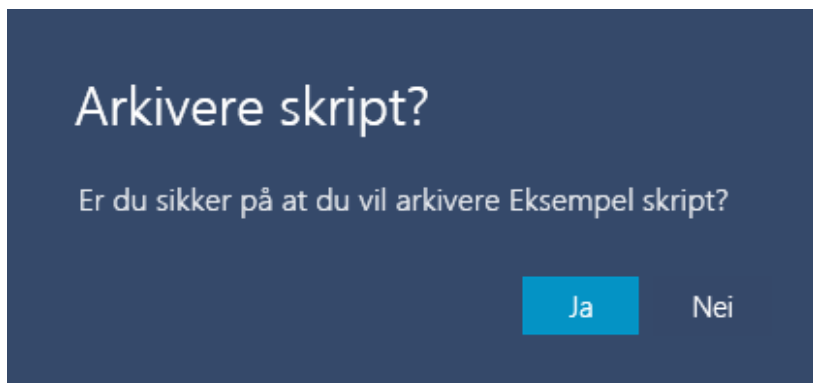
Skript som ikke lenger skal benyttes kan arkiveres. For å arkivere et skript kan du enten bruke søppelbøtte-ikonet i menylinjen i skripteditoren, eller du kan høyreklikke på skriptnavnet og trykke «Arkiver skript». Bekreft arkiveringen ved å klikke på «Ja».

Arkiverte skript vises i en liste i skjermbildet.

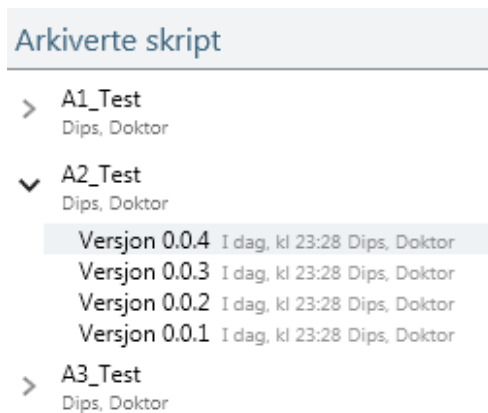
Du kan se alle versjoner av et skript også når det er arkivert.



Figur 26. Arkivere skript



Figur 27. Bekrefte arkivering skript



Figur 28. Liste over arkiverte skript

1.2.13.1. Skjermbildereferanser

Skjermelement	Beskrivelse
Arkiver	Knappen for arkivering av det aktuelle skriptet du har åpnet i skripteditoren.

Tabell 10. Skjermfeltbeskrivelser

1.2.14. Gjenopprette skript

Dersom du ønsker ta i bruk et arkivert skript, kan du bruke funksjonen for gjenoppretting. Det vil si at et arkivert skript kan aktiveres igjen.

Når et skript arkiveres fra skriptlisten i flisen, slettes ikke skriptet i databasen. Alle skript som lagres i flisen blir tatt vare på ved arkivering, og vises i listen over arkiverte skript.

Dersom du ønsker ta i bruk igjen et arkivert skript, gjøres dette ved å trykke på ønsket skript og deretter trykke på knapp for *Gjenoprett skript*.

Når du trykker på knappen aktiveres skriptet, og legger seg i listen over aktiverte skript.

Merk at dersom skriptet er en eldre versjon som ikke lenger støttes i gjeldende Arena versjon, må du først redigere skriptet før det kan lagres som aktivt skript.



Figur 29. Menyknapp for gjenopprette skript

1.2.14.1. Skjermbildereferanser

Skjermelement	Beskrivelse
Gjenoprett skript	Knappen for gjenoppretting av det arkiverte skriptet du har åpnet i skripteditoren.

Tabell 11. Skjermfeltbeskrivelser

:anchor

arenaworkflowskript

1.3. Oppsett, konfigurasjon og tilgangskontroll

1.3.1. Forutsetninger for bruk

Følgende produkter og lisenser er påkrevd:

Produkt-/lisensnavn	Produktnr.	Lisens id.
DIPS Arena Process Manager	1512	1823

Tabell 12. Påkrevde produkter/lisenser

Personer som skal bruke editoren i Arbeidsflytskriptflisen må ha inngående kjennskap til systemet og konfigurasjon generelt, arketyper og ha erfaring med programmering (C#).

1.3.2. Tilgangskontroll

For generell beskrivelse av tilgangskontroll, forutsettes det at dokumentet [Tilgangskontroll](#) leses. Nedenfor beskrives oppsett av tilgangskontroll for flisen «Arbeidsflytskript» .

For å få tilgang til «Arbeidsflytskript» i Arena må elementtypen «Arena Admin, Arbeidsflytskript - Vis flisa Arbeidsflytskript» være aktivert for brukerrollen. Se tabell [Elementtyper](#)

Det er ikke implementert adgangskontroll på øvrig innhold (skript og funksjoner) i flisen. Dersom du har tilgang til flisen, har du også tilgang til alle skript og funksjoner i flisen.

1.3.2.1. Elementtyper

Elementtypeld	Elementtype	Beskrivelse
4421	Arena Admin, Arbeidsflytskript - Vis flisa Arbeidsflytskript	Elementtypen gir tilgang til flisen «Arbeidsflytskript».

Tabell 13. Oversikt elementtyper for Arbeidsflytskript

