3 Dokumentasjon av fullstendig API

HarbFramework APIet er definert i 5 forskjellige Interface. Interfacene definerer kontrakten for klassene og metodene som skal være tilgjengelig for brukere av APIet. Brukeren kan opprette objekter av klassene som er tilknyttet hvert av de forskjellige API interfacene og bruke metodene som interfacene definerer. Klassene vil ha flere underliggende metoder en det interfacene definerer, men disse metodene er ikke tilgjengelig gjennom APIet og ikke metoder brukeren skal trenge å forholde seg til.

IEvent

IEvent definerer APIet for event objekter. Disse objektene holder informasjon om historikken til skip og kontainere som brukes i simuleringen. Hvert objekt holder informasjon om statusen til ett skip eller kontainer på ett gitt tidspunkt. Objektene opprettes hver gang en kontainer eller skip endrer status (arbeidsoppgave) fra en oppgave til en annen. For eksempel vil ett event objekt bli opprettet når ett skip går fra å laste av kontainere (Unloading) til å laste på nye kontainere (Loading). Disse event objektene vil bli lagt til hvert enkelt skips eller kontainers historie slik at de kan hentes ut for å få en oversikt over hva skipet eller kontaineren har gjort gjennom forløpet til simuleringen.

IEvent definerer følgende metoder:

public Guid Subject { get; }

Denne metoden returnerer Guid til den spesifikke kontaineren eller skipet som eventen er knyttet til. Hvert skip og kontainer har en unik ID som identifiserer det skipet eller kontaineren. Variabelen subjekt identifiserer dette skipet eller kontaineren slik at vi kan vite hvilke skip eller kontainer eventen handler om.

public Guid SubjectLocation { get; }

Denne metoden returnerer Guid til lokasjonen som Subjectet til eventen befinner seg i det øyeblikket event objektet ble opprettet. SubjectLocation kan for eksempel være skipet som en kontainer befinner seg på. En havnplass som ett skip befinner seg på eller en kontainerplass på havna som en kontainer befinner seg på. De forskjellige lokasjonene er som følger:
For skip: Anchorage, Transit, LoadingDock, ShipDock.
For containere: Ship, ContainerSpace.

Alle disse forskjellige lokasjonene har sin egen ID. Hvert unike skip, loading dock, ShipDock vil ha sin egen unike ID.

public DateTime PointInTime { get; }

Denne metoden returnerer ett DateTime objekt med tidspunktet som eventen ble opprettet. Eventer opprettes når Subjectet de omhandler endrer status. For eksempel hvis en kontainer

går fra å bli lastet av ett skip (Unloading) til å bli lagret på et containerspace på havnen (InStorage). Du kan dermed se ut ifra PointInTime variabelen og Statusen når Subjectet fikk sin nåværende status.

public Status Status { get; }

Denne metoden returnerer en Status enum som beskriver statusen Subjectet har fått idet Event objektet opprettes. Status enumen gir informasjon om hva som skjer med, eller hva subjectet gjør i det øyeblikket Event objektet blir opprettet. Statusene som ett subject kan ha er som følger:

For ship:

- DockingToLoadingDock (Skipet legger til havn ved en losseplass hvor skipet kan laste av og på kontainere)

- DockingToShipDock (Skipet legger til havn ved en vanlig kaiplass)

- Undocking (Skipet legger fra havn)

- Loading (Skipet laster kontainere fra havnen til sitt eget lager)

- Unloading (Skipet lasterkontainere fra sitt eget lager til havnen)

- Transit (Skipet er i transit ute på havet)

- Anchoring (Skipet ligger til anker hvor det venter på å få en ledig kaiplass)

For container:
- Loading (Kontaineren blir lastet ombord på ett skip)
- Unloading (Kontaineren blir lastet av ett skip og til en ledig containerSpace på havnen)
- Transit (Kontaineren er ombord på ett skip som er i transit på havet)

- InStorage (Kontaineren er lagret på en ContainerSpace på havnen)

IContainer

IContainer definerer APIet for Container klassen. Objekter av denne klassen holder representerer en container i simuleringen og holder på relevant informasjon om denne kontaineren slik som IDen til kontaineren, historikken til kontaineren, størrelsen på containeren, vekten til kontaineren og containerens nåværende lokasjon. Objektene opprettes typisk i skips lasterom samtidig som nye skip opprettes.

Det er vert å merke seg at konstruktøren for container klassen ikke er tilgjengelig for brukere av APIet, istede opprettes containere samtidig med opprettelse av skip. Dette er for å minimere arbeidet brukere må gjøre for å kunne starte en ny simulering.

Følgende metoder er gitt i IContainer interfacet:

public Guid ID { get; }

Denne metoden returnerer ett Guid objekt som holder på den unike IDen til containeren. IDen kan brukes til å identifisere og finne spesifikke kontainere i andre metodekall.

public IList<Event> History { get; }

Denne metoden returnerer en IList med Event objekter (Se beskrivelse av IEvent for nærmere forklaring av disse objektene). Denne listen inneholder ett Event objekt for hver gang kontaineren har endret Status (se beskrivelse av Status enum). Listen er organisert fra første status endring til siste, der første statusendring har index 0 i listen og den siste status endringen har den siste indexen i listen. Man kan slik få en komplett oversikt over hva en container har gjort og hvor den har befunnet seg til enhver tid gjennom hele simuleringen.

public ContainerSize Size { get; }

Denne metoden returnerer en ContainerSize enum (se beskrivelse av Container Size enum for nærmere informasjon) som definerer størrelsen på kontaineren. Kontainerens vekt er gitt ut ifra størrelsen og forskjellige container størrelser krever forskjellig størrelse på lagerplassen den kan få tildelt på havnen.

public int WeightInTonn { get; }

Denne metoden returnerer en int med vekten til containeren gitt i tonn. Vekten på en kontainer kan variere basert på størrelsen og skip har en samlet maksvekt på kontainere de kan laste ombord.

public Guid CurrentPosition { get; }

Denne metoden returnerer ett Guid objekt som holder på IDen til objektet som i dette øyeblikket lagrer containeren. Dette kan være IDen til ett skip kontaineren er lagret ombord hos, eller det kan være en lagerplass på havnen som kontaineren er lagret på mens den venter på at ett skip skal laste den ombord.

public Status GetStatus();

Denne metoden returnerer en status enum (Se enum status for mer informasjon) som indikerer statusen til skipet på det tidspunktet metoden blir kalt.

IShip

IShip definerer APIet til skip objekter. Ett skip objekt definerer ett enkelt skip som skal brukes i simuleringen. Alle skip i simuleringen er lasteskip som kan laste kontainere. Skip kan komme i flere størrelser (se ShipSize enum for mer informasjon). Størrelsen til skipet vil

definere maksvekten av lasten ett skip kan ha ombord, antall kontainere den har plass til i lageret sitt og hvilke størrelse på havneplass skipet krever for å kunne legge til kai.

Skip kan enten opprettes for en enkeltseilaser eller for kontinuerlige seilaser. For enkeltseilaser vil skipet ankomme havnen ved en gitt dato, legge seg til en ledig lastehavn (loading dock), laste av lasten sin for deretter og flytte seg til en vanlig skipshavn (ship dock). Der vil den ligge ut simuleringen.

For ett skip med kontinuerlig seilas kan det settes antall dager det tar fra ett skip drar fra kai til skipet er tilbake igjen ved havnen. Skipet vil deretter legge seg til en ledig lastehavn (loading dock), laste av lasten sin, laste på ny last for deretter og legge fra kai og dra ut på havet igjen. Skipet vil repetere denne syklusen for hele lengden av simuleringen.

public Ship (ShipSize shipSize, DateTime StartDate, bool IsForASingleTrip, int roundTripInDays, int numberOfcontainersOnBoard)

Denne konstruktøren tar imot 5 attributter:

- **shipSize:** tar imot en enum ShipSize (se seksjon om Enum Shipsize for mer informasjon) som bestemmer størrelsen på skipet. Størrelsen på skipet vil definere maksvekten på den totale lasten ett skip kan ha ombord og antallet kontainere ett skip har plass til i lagerrommet, i tilleg til hvor stor kaiplass skipet trenger når det legger til kai.

- **startDate**: tar imot ett DateTime objekt som definerer tidspunktet hvor skipet først ankommer havnen. Skipet vil ankomme havnen for å laste av sin første last på datoen gitt i attributtet.

- **IsForASingleTrip:** Tar imot en boolsk verdi (true/false) som sier om skipet er et enkeltseiling-skip eller ikke.

- **roundTripInDays:** tar imot en int verdi som definerer hvor mange dager det tar fra ett skip legger fra kai, drar ut på havet til det er tilbake igjen med ny last.

- **numberOfcontainersOnBoard:** tar imot en int verdi og definerer hvor mange kontainere skipet har ombord første gangen skipet ankommer havnen. Denne attributtene definerer altså ikke maks antall kontainere ett skip kan ha ombord ettersom dette defineres av skipets størrelse, men antallet containere som skipet har i lasterommet ved første ankomst til havnen. Ved opprettelse av skip vil programmet forsøke å generere ett nogen lunde likt antall kontainere av hver størrelse i skipets last. Man kan dermed forvente at hvis man sender inn en verdi av 3 til denne attributtene vil skipet starte med 1 liten kontainer, 1 medium kontainer og 1 stor kontainer ombord.

I tillegg er disse metodene gitt i IShip interfacet:

public Guid ID { get; }

Denne metoden returnerer et Guid objekt som holder på den unike IDen til skipet. Denne IDen kan brukes til å identifisere det spesifikke skipet i andre metodekall.

public ShipSize ShipSize { get; }

Denne metoden returnerer ett ShipSize enum (se Enum Shipsize for mer informasjon) som definerer størrelsen på skipet. Størrelsen på skipet vil definere maksvekten på den totale lasten ett skip kan ha ombord og antallet kontainere ett skip har plass til i lagerrommet, i tilleg til hvor stor kaiplass skipet trenger når det legger til kai.

public DateTime StartDate { get; }

Denne metoden returnerer ett DateTime objekt som gir tidspunktet for første gang skipet ankom havnen.

public int RoundTripInDays { get; }

Denne metoden returnerer en int verdi som sier hvor mange dager det tar fra skipet legger fra kaien med ny last til den er tilbake igjen til havnen. Altså tiden det tar for skipet å dra ut på havet for så å komme tilbake til havnen igjen.

public Guid CurrentLocation { get; }

Denne metoden returnerer Guid til den nåværende possisjonen til skipet. Possisjonene ett skip kan befinne seg er:

- Transit: Skipet er på havet.
- Anchorage: Skipet ligger til anker og venter på å få en ledig kaiplass til å kunne laste av lasten sin-

- LoadingDock: skipet ligger til en laste kai hvor den laster av og/eller på kontainere. Hver enkelt loadingDock har sin unike Guid.

- ShipDock: skipet er ferdig med enkelt-seilasen sin og ligger nå til kai. Hver enkelt shipDock har sin unike Guid.

public IList<Event> History { get; }

Denne metoden returnerer en IList med Event objekter (Se beskrivelse av IEvent for nermere forklaring av disse objektene). Denne listen inneholder ett Event objekt for hver gang skipet har endret Status (se beskrivelse av Status enum). Listen er organisert fra første status endring til siste, der første statusendring har index 0 i listen og den siste status endringen har den siste indexen i listen. Man kan slik få en komplett oversikt over hva en skipetr har gjort og hvor det har befunnet seg til enhver tid gjennom hele simuleringen.

public IList<Container> ContainersOnBoard { get; }

Denne metoden returnerer en IList med Container objekter som representerer containerene som skipet har i sitt lasterom i nåværende øyeblikk.

public int ContainerCapacity { get; }

Denne metoden returnerer en int verdi som representerer hvor mange containere skipet har plass til i lasterommet sitt. En kontainer vil ta opp 1 plass i lasterommet uavhengig av størrelsen på kontaineren.

public int MaxWeightInTonn { get; }

Denne metoden returnerer en int verdi som representerer hva maksvekten i tonn skipet kan være før den synker. Denne maksvekten inkluderer vekten på selve skipet pluss vekten på lasten som skipet har ombord. Hvor stor maksvekt ett skip kan ha er basert på skipets størrelse. Større skip vil ha en større maksvekt. Maksvekten til skipet kan ikke overskrides.

public int BaseWeightInTonn { get; }

Denne metoden returnerer en int verdi som representerer vekten i tonn skipet er når den ikke har noen last ombord. Altså når lasterommet er tomt. Denne vekten medgår i den nåværende vekten til skipet (CurrentWeightInTonn). Base vekten til skipet er basert på størrelsen til skipet. Større skip vil ha en større dødvekt.

public int CurrentWeightInTonn { get; }

Denne metoden returnerer en int verdi som representerer vekten i tonn som skipet i øyeblikket metoden kalles. Vekten kalkuleres med grunnlag på vekten skipet har når det er tomt (BaseWeightInTonn) + summen av vekten til alle kontainerne som skipet har i sitt lasterom. Dette tallet kan aldri overskride maksvekten skipet kan ha (MaxWeightInTonn).

IHarbor

IHarbor definerer APIet for harbor objekter. Ett harbor objekt representerer en enkelt havn som kan brukes i simuleringen. Hver simulering bruker kun en havn. Harbor er objektet som driver det meste av det som skjer i simuleringen. Den holder på informasjon om skipene som brukes i simuleringen og hvor skipene befinner seg. Den har også lastekaiplasser (loadingDocks), kaiplasser der skip kan ligge til havn over lengere perioder (ShipDocks) og lagerplasser for kontainere (ContainerSpaces). Skip vil ankomme havnen og laste av og på kontainere før de drat ut på havet igjen. Harbor objektet holder på all denne informasjonen om hvilke skip som er med i simuleringen og hvor de befinner seg.

Harbor objekter har en public konstruktør som ikke er gitt i interfacet. Denne konstruktøren ser slik ut:

public Harbor(IList<Ship> listOfShips, int numberOfSmallLoadingDocks, int numberOfMediumLoadingDocks, int numberOfLargeLoadingDocks, nt numberOfSmallShipDocks, int numberOfMediumShipDocks, int numberOfLargeShipDocks,

int numberOfSmallContainerSpaces, int numberOfMediumContainerSpaces, int numberOfLargeContainerSpaces)

Denne konstruktøren tar 10 attributter som er som følger:

- listOfShips: Denne attributtene er en liste med Ship objekter av type IList<Ship> som skal brukes i simuleringen. Alle skip vil starte på ankerplassen mens de venter på å få komme til en ledig kaiplass for å kunne laste av lasten sin.

- **numberOfSmallLoadingDocks:** Denne attributtene tar en int verdi som angir hvor mange laste-kai-plasser som havnen har tilgjengelig for sip av størrelse Smal (se Enum Shipsize for mer info). LoadingDocks eller laste-kai-plasser er kaiplasser der skip kan legge til og laste av og på kontainere før de drar ut på havet igjen. Hver LoadingDock kan ta ett skip av tilsvarende størrelse som seg selv.

- **numberOfMediumLoadingDocks:** Tilsvarende numberOfSmallLoadingDocks bare for loading docks av medium skipstørrelse i stedet.

- **numberOfLargeLoadingDocks:** Tilsvarende numberOfSmallLoadingDocks bare for loading docks av Large skipstørrelse i stedet.

- **numberOfSmallShipDocks:** Denne attributten tar en int verdi som angir hvor mange kaiplasser havnen skal ha tilgjengelig for skip av størrelse Small (see Enum ship size for mer info) for å kunne ligge til kai når de ikke laster av og på kontainere. Det er typisk sett skip som bare gjør en enkeltseilas som benytter seg av denne typen kaiplasser. Ett skip av denne typen vil legge seg til en ledig ShipDock som matcher sin egen størrelse når den er ferdig med å laste av lasten sin. Der vil skipet ligge til simuleringen er ferdig.

- **numberOfMediumShipDocks:** Tilsvarende numberOfSmallShipDocks bare for docks av størelse medium istede.

- **numberOfLargerShipDocks:** Tilsvarende numberOfSmallShipDocks bare for docks av størelse large istede.

- **numberOfSmallContainerSpaces:** Denne attributten tar imot en int verdi som definerer hvor mange lagerplasser (container space) for kontainere av størrelse Small (see Enum ContainerSize for mer info) havnen skal ha tilgjengelig. En containerSpace kan lagre en kontainer av tilsvarende størelse som seg selv. For eksempel kan en smal container space lagre en smal container.

- **numberOfMediumContainerSpaces:** tilsvarende numberOfSmallContainerSpaces bare for Container Spaces av størrelse medium i stedet.

- **numberOfLargeContainerSpaces:** tilsvarende numberOfSmallContainerSpaces bare for Container Spaces av størrelse large i stedet.

I tilleg er disse metodene gitt i interfacet:

public Guid ID { get; }

Denne metoden returnerer ett Guid objekt som definerer den unike IDen til Harbor objektet. IDen kan brukes for å identifisere og skille harbor objekter fra hverandre.

public IDictionary<Guid, bool> LoadingDockIsFreeForAllDocks();

Denne metoden returnerer en liste sortert i key-value par der Key er ett Guid objekt som representerer den unike IDen til en enkel laste-kai-plass (loading dock), og value er en bool verdi som har verdien True hvis laste-kai-plassen er ledig og False hvis den er opptatt. Listen inneholder slike key-value par for alle laste-kai plassene på havnen og kan brukes for å få en oversikt over hvilke kaiplasser som er ledig eller opptatt på tidspunktet metoden blir kalt.

public bool LoadingDockIsFree(Guid dockID);

Denne metoden tar imot en Guid til en LoadingDock (laste-kai-plass) og returnerer en bool verdi som er true hvis kaiplassen er ledig og false hvis den er opptatt. Den kan dermed brukes for å sjekke ledigheten til en enkelt laste-kai-plass.

public Status GetShipStatus(Guid ShipID);

Denne metoden tar imot ett Guid objekt med IDen til ett enkelt ship objekt og returnerer en Status enum (se enum ship status for mer informasjon) som representerer statusen skipet har når metoden blir kalt. Ett skip kan ha følgende statuser:
- Transit: Skipet er på havet.
- Anchorage: Skipet ligger til anker og venter på å få en ledig kaiplass til å kunne laste av lasten sin-

- LoadingDock: skipet ligger til en lastekai hvor den laster av og/eller på kontainere. Hver enkelt loadingDock har sin unike Guid.

- ShipDock: skipet er ferdig med enkelt-seilasen sin og ligger nå til kai. Hver enkelt shipDock har sin unike Guid.

public IDictionary<Guid, bool> ShipDockIsFreeForAllDocks();

Denne metoden returnerer en value-key liste av typen IDictionary<Guid, bool> der key verdiene er objekter av typen Guid som representerer de unike IDene til hver enkelt shipDock (Kaiplass der skip ligger til kai uten å kunne laste av og på containere) og valuene er bool verdier som er true hvis docken er ledig og False hvis den er opptatt.

public IDictionary<Ship, Status> GetStatusAllShips();

Denne metoden returnerer en key-value liste av typen IDictionary<Ship, Status> der key verdien er ett Ship objekt og value verdien er en Status enum (se Enum status for mer info) som beskriver statusen skipet har i øyeblikket metoden blir kalt. Man kan på denne måten få oversikt over hva hvert enkelt skip i simuleringen gjør på tidspunktet metoden blir kalt.

public Guid AnchorageID { get; }

Denne metoden returnerer ett Guid objekt som holder på den unike IDen til Anchorage. Anchorage er en posisjon som skip kan befinne seg i. Den representerer når skip ligger til anker og venter på å få en ledig kaiplass ved havnen.

public Guid TransitLocationID { get; }

Denne metoden returnerer ett Guid objekt som holder på den unike IDen til Transit lokasjonen. Transit er en posisjon som ett skip kan befinne seg i. Den representerer når skip er på havet.

ILog

ILog definerer APIet til Log klassen. Log klassen kan brukes til å lage Log objekter som simuleringen bruker for å bokføre statusen til alle skip og kontainere i simuleringen på ett gitt tidspunkt. Simuleringen vil hvert døgn opprette en nytt Log objekt som forteller om status og plassering som alle skip og containere har i det øyeblikket logobjektet ble opprettet.

ILog interfacet definerer følgende metoder:

public DateTime Time { get; }

Denne metoden returnerer ett DateTime objekt som angir tidspunktet i simuleringen logobjektet ble opprettet. Denne verdien angir altså tidspunktet som alle skip og containere hadde status og plassering som er angitt i log objektet.

public IList<Ship> ShipsInAnchorage { get; }

Denne metoden returnerer en liste av type IList<Ship> som inneholder alle skip som lå til anker på det tidspunktet hvor Log objektet ble opprettet.

public IList<Ship> ShipsInTransit { get; }

Denne metoden returnerer en liste av type IList<Ship> som inneholder alle skip som var på havet på det tidspunktet Log objektet ble opprettet.

public IList<Ship> ShipsDockedInShippingDocks { get; }

Denne metoden returnerer en liste av type IList<Ship> som inneholder alle skip som lå til en laste-kai (Loading dock) og lastet av og på containere på det tidspunktet Log objektet ble opprettet.

public IList<Ship> ShipsDockedInShippingDocks { get; }

Denne metoden returnerer en liste av type IList<Ship> som inneholder alle skip som lå til en kaiplass for skip som ikke laster av eller på containere på det tidspunktet Log objektet ble opprettet

public IList<Container> ContainersInHarbour { get; }

Denne metoden returnerer en liste av type IList<Container> som inneholder alle containere som lå lagret til land på havnen i på det tidspunktet Log objektet ble opprettet.

public void PrintInfoForAllShips();

Denne metoden printer informasjon om alle skip og hvor de befinner seg til konsoll.

public void PrintInfoForAllContainers);

Denne metoden skriver ut informasjon om alle containere og hvor de befinner seg til konsoll.

ISimulation

ISimulation definerer APIet til Simulation klassen. Simulation klassen kan brukes for å lage Simulation objekter som brukes for å kjøre simuleringen til en havn.

Simulation klassen har en public konstruktør som ikke er definert i interfacet. Denne konstruktøren ser slik ut:

public Simulation (Harbor harbor, DateTime simulationStartTime, DateTime simulationEndTime)

Konstruktøren har 3 attributter:

- Harbor: tar inn et Harbor objekt som er havnen som skal brukes i simuleringen.

- simulationStartTime: tar imot ett DateTime objekt som angir tidspunktet som er første dag simuleringen skal starte på.

- simulationEndTime: tar imot ett DateTime objekt som angir tidspunktet som simuleringen skal kjøre til. Altså tidspunktet simuleringen skal stoppe.

I tillegg angir interfacet følgende metoder:

public IList<Log> History { get; }

Denne metoden returnerer en IList<Log> med log objekter (Se ILog for mer informasjon om disse). Denne listen inneholder ett Log objekt for hver dag simuleringen har pågått. Nye logg objekter opprettes og legges til listen klokken 12 om natten hver 24 time. Listen er organisert sekvensielt slik at det første log objektet som blir opprettet ligger først på index 0 og det siste log objektet som ble opprettet ligger på den høyeste indexen i listen. Man kan slik ved hjelp av denne listen få en komplett oversikt over posisjonen og statusen til alle skip og kontainere gjennom hele forløpet til simuleringen.

public IList<Log> Run();

Denne metoden brukes til å starte simuleringen. Når metoden kalles vil simuleringen starte og kjøre fra tidspunktet gitt i konstruktørens start time og til tidspunktet gitt i konstruktørens end time. Den returnerer så en IList<Log> med log objekter som forteller om historien til hvordan simuleringen har utspilt seg.

public void PrintShipHistory();

Denne metoden printer til konsoll hele historien til alle skip for den forrige simuleringen som har blitt utført.

public void PrintContainerHistory();

Denne metoden printer til konsoll hele historien til alle kontainere for den forige simuleringen som har blitt utført.

Enum ContainerSize

Enumen ContainerSize er tilgjengelig gjennom APIet. Enumen angir størrelsen til en container. Alle containere har en container size enum som representerer størrelsen kontaineren kan ha. Størrelsene er som følger:

None = 0,

Small = 10,

Medium = 15,

Large = 20

Verdiene gitt etter hver av størrelsene er vekten på containeren i tonn. For eksempel vil en Large container veie 20 tonn.

Enum ShipSize

Enumen ShipSize er tilgjengelig gjennom APIet. Enumen angir størrelsene skip kan ha. Alle skip har en ShipSize enum som definerer størrelsen til skipet. Størrelsene er som følger:

None = 0,

Small,

Medium,

Large

Enum Status

Enumen Status er tilgjengelig gjenom APIet. Enumen angir statuser som skip og kontainere kan ha gjennom simuleringen. Hver status representerer noe ett skip eller kontainer kan gjøre på ett gitt tidspunkt. Statusene er som følger:

None = 0

DockingToLoadingDock - skip legger seg til en laste-kai

DockingToShipDock - skip legger seg til en kai hvor de ikke kan laste.

Undocking - skip legger fra kai

Loading - skip laster ombord containere eller en container lastes ombord på ett skip.

Unloading - skip laster av containere eller en container lastes av ett skip.

Transit - skip er på havet.

InStorage - container er lagret på land på havna

Anchoring - skip ligger til anker og venter på en ledig kaiplass

UnloadingDone - skip er ferdig med å laste av lasten sin.

LoadingDone - skip er ferdig med å laste på ny last