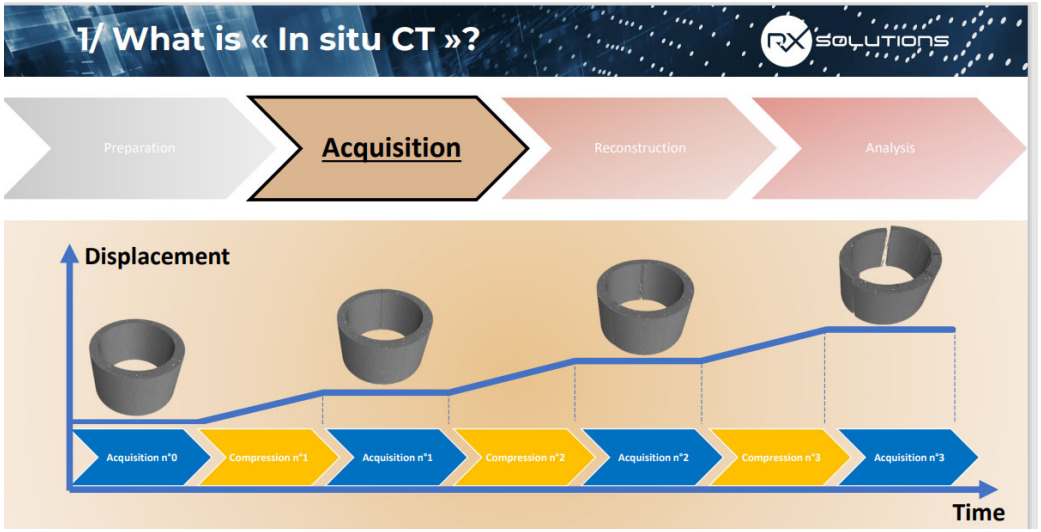
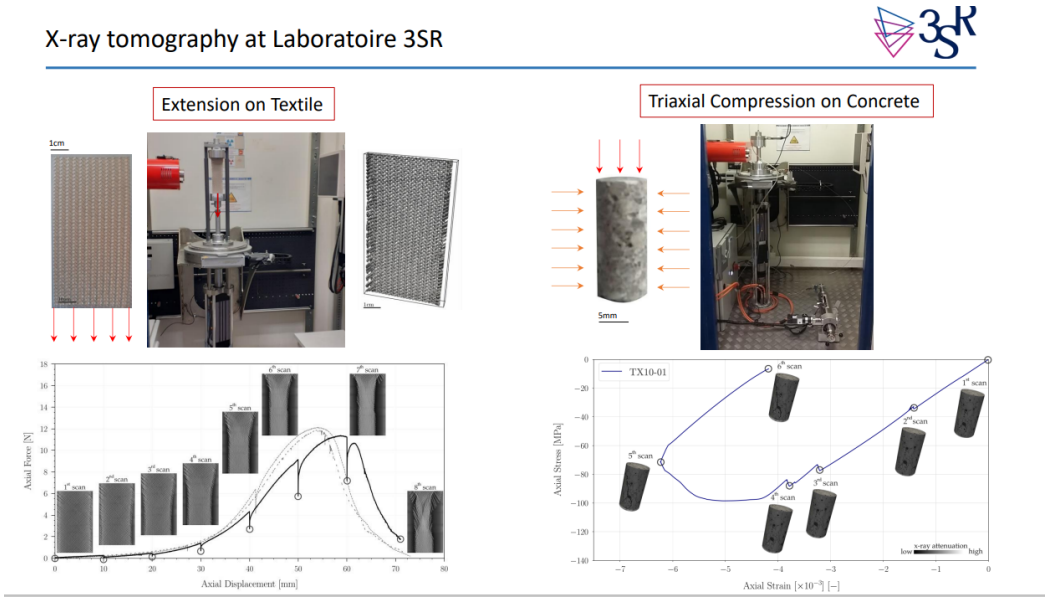
Journée Alpview
14 novembre 2024

« Utilisation d'une API pour automatiser un test in situ sous Rayons X »

Pascal charrier
Laboratoire 3SR,
Domaine Universitaire Grenoble





thanks@Olga Stamati

Temps d'acquisition classique matériau : 90 min , 1200 radiographies

Durée « mécanique » : 15 min
Chargement + temps de relaxation

Temps test classique : une dizaine de cycle : (90 +15) x 10

Etablir le dialogue entre l'application qui gère le chargement (mécanique) et l'application qui gère l'acquisition des radiographies tomographie entre chaque étape de chargement

Moteur & Pompes / Capteurs / Paramètres de Chargement Mécanique ⟷ Générateur X / Détecteur / Paramètres d'Acquisition

7h30-20h sur place → 20h-7h30 & WE en remote access → Pour la nuit et WE

API c’est quoi et pourquoi ?

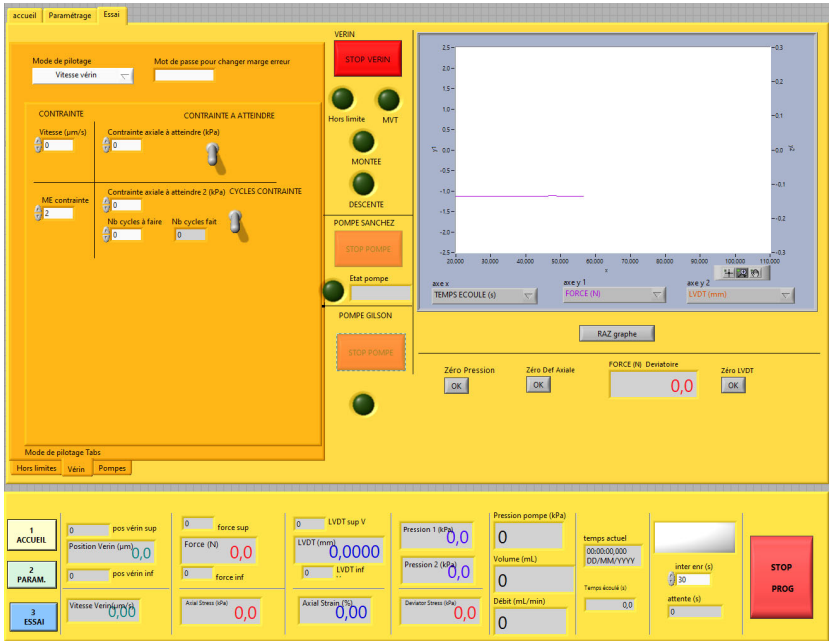
- **Fonction principale d'une API**

- L'idée principale derrière une API est de permettre à 2 systèmes de s'échanger des informations sans que l'utilisateur ait besoin d'interagir directement avec le système. L'API agit comme un intermédiaire entre 2 parties, en facilitant l'accès aux fonctionnalités ou aux données d'un service, d'une application ou d'un système

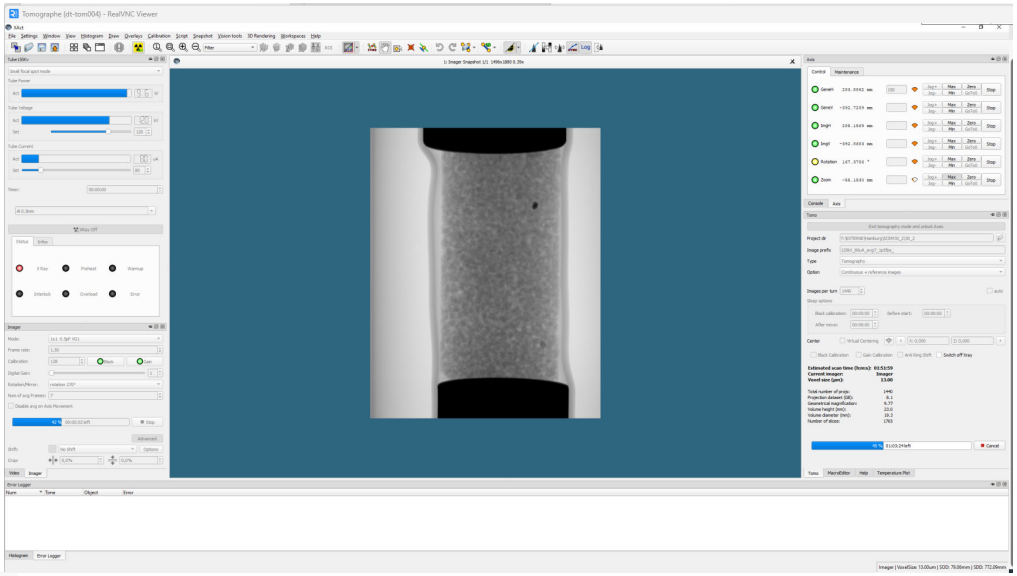
- Il existe différents types d'API en fonction du contexte d'utilisation :

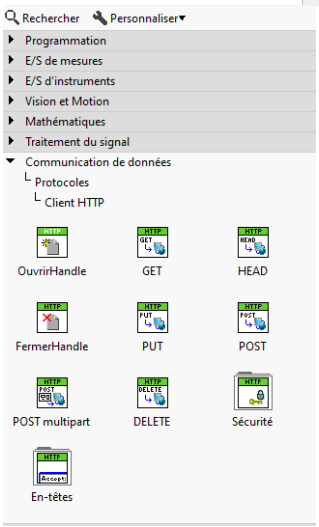
- **API Web** : Ce sont les plus courantes et **elles permettent aux applications de communiquer** via internet, par exemple une API REST qui permet à une application de récupérer des données depuis un serveur distant.

Ce celle-ci que l’on utilise dans notre cas fournie par le fournisseur du Logiciel Xact qui pilote la partie tomographie


accueil
Paramétrage
Essai
Mode de pilotage
Vitesse vérin
Mot de passe pour changer marge erreur
CONTRAINTE
CONTRAINTE A ATTEINDRE
Vitesse (µm/s)
Contrainte axiale à atteindre (kPa)
ME contrainte
Contrainte axiale à atteindre 2 (kPa)
CYCLES CONTRAINTE
Nb cycles à faire
Nb cycles fait
VERIN
STOP VERIN
Hors limite
MVT
MONTEE
DESCENTE
POMPE SANCHEZ
STOP POMPE
Etat pompe
POMPE GILSON
STOP POMPE
RAZ graphe
axe x
TEMPS ECOULE (s)
axe y 1
FORCE (N)
axe y 2
LVDT (mm)
Zéro Pression
Zéro Def Axiale
FORCE (N) Deviatoire
Zéro LVDT
OK
Mode de pilotage Tabs
Hors limites
Vérin
Pompes
1 ACCUEIL
2 PARAM.
3 ESSAI
pos vérin sup
Position Verin (µm)
pos vérin inf
Vitesse Verin (µm/s)
force sup
Force (N)
force inf
LVDT sup V
LVDT (mm)
LVDT inf
Axial Stress (kPa)
Axial Strain (%)
Pression 1 (kPa)
Pression 2 (kPa)
Deviator Stress (kPa)
Pression pompe (kPa)
Volume (mL)
Débit (mL/min)
temps actuel
Temps écoulé (s)
inter enr (s)
attente (s)
STOP PROG

API




Rechercher
Personnaliser
Programmation
E/S de mesures
E/S d'instruments
Vision et Motion
Mathématiques
Traitement du signal
Communication de données
Protocoles
Client HTTP
OuvrirHandle
GET
HEAD
FermerHandle
PUT
POST
POST multipart
DELETE
Sécurité
En-têtes

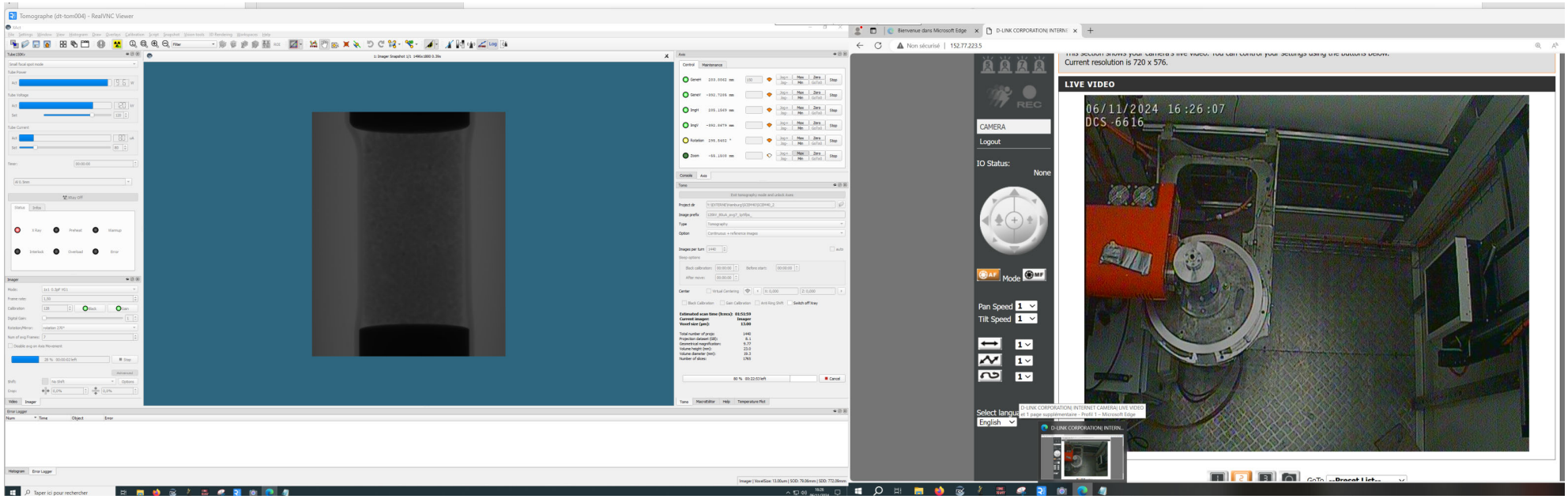
API de Xact

REST API – Organization

- Each object can be controled through attributes, commands and requests:
  - The **attributes** correspond to the parameters of the machine
    - *Example : http://localhost/api/Imager/attributes/avrFrames*
  - The **commands** correspond to actions on the machine
    - *Example : http://localhost/api/Imager/command/start*
  - The **requests** permit:
    - to retrieve the status and errors of each object
    - some special operations to start a live imager broadcasting or to get reconstructed data

API operations	HTTP		Websocket
	GET	POST	
Commands		Execute an action on the machine	
Attributes	Return the value of the parameter	Change the value of the parameter	
Requests	Return the status and the errors of the machine or object		Imager streaming / UniCT data retriever

Interface du module Xact qui pilote la partie Acquisition des Radiographies



- C’est ce soft qui est le cote serveur de l’API

Permission Token

2.4. Permission token to modify the resources

Since concurrent access to modify the machine may happen and lead to hazardous behaviour, two special commands have to be used to control the machine, **hold** and **release**. The first command requests a permission token that allows the client to change the state of the machine. The second command releases the token once the client has finished to modify the machine state.
Note: only one token can be provided at the same time. The python script below shows an example of how to get and release such token.

Hold a permission token

Python script

```
import requests
import json

response = requests.get("https://XACT_SERVER_ADDRESS/API/hold")
json_response = json.loads(response.text)
```

Content of the json_response variable

```
{
    "object": "",
    "operation": "hold",
    "response":
    {
        "type": "message",
        "message": "07d438945d16459daa7b22be7dd8e23b"
    }
}
```

Relase a permission token

Python script

```
import requests
import json

post_body = {"permission":"07d438945d16459daa7b22be7dd8e23b", "parameters":[]}
response = requests.post("https://XACT_SERVER_ADDRESS/API/release", json = post_body)
json_response = json.loads(response.text)
```

Content of the json_response variable

```
{
    "object": "",
    "operation": "release",
    "response":
    {
        "type": "message",
        "message": "The machine has been successfuly released"
    }
}
```

1. Introduction

This API is a web service that permits to access and program RX Solutions X-Act software to perform a well-defined sequence of actions. This web service proposes several ways to communicate with X-Act:

- GET method to get the current state of the machine (cf. sections 2 and 3),
- POST method to change the state of the machine (cf. sections 2 and 3),
- WebSocket protocol to stream imager live or to receive reconstructed data (cf. section 4).

There are several resources available on the machine (xRay sources, imagers, axes, ...) listed on the left menu. For each ressouce a detailed list of the available attributes, commands and requests is provided.

2. Special commands

2.1. List all available resources and all available attributes, commands and requests of a resource

It is possible to have a list of all available resources that can be controlled through the API, and, for each resource, it is also possible to obtain a list of all attributes, commands and requests. The python script below shows an example of how to get these lists.

Get all available resources

Python script

```
import requests
import json

response = requests.get("https://XACT_SERVER_ADDRESS/API")
json_response = json.loads(response.text)
```

Content of the json_response variable


```
{
    "object": "",
    "operation": "requestAvailableObjects",
    "response":
    {
        "message":
        {
            "Axis": ["Horizontal", "ImagerH", "ImagerV", "ImagerZoom", "Rotation", "Vertical", "Zoom"],
            "Imager": ["Imager", "Lucam"], "Io": ["Io"],
            "Misc": ["Tomography", "INach3", "KeepOut", "Laminography", "Automation", "BeckhoffTemperatureSensor", "DisplayZoom", "Ge
            "Source": ["Xray150"],
            "TouchLcd": ["ILcd"]
        },
        "type": "message"
    }
}
```


Get available attributes, commands and requests for Xray150 resource

Python script

```
import requests
import json

response = requests.get("https://XACT_SERVER_ADDRESS/API/Xray150")
json_response = json.loads(response.text)
```

Content of the json_response variable

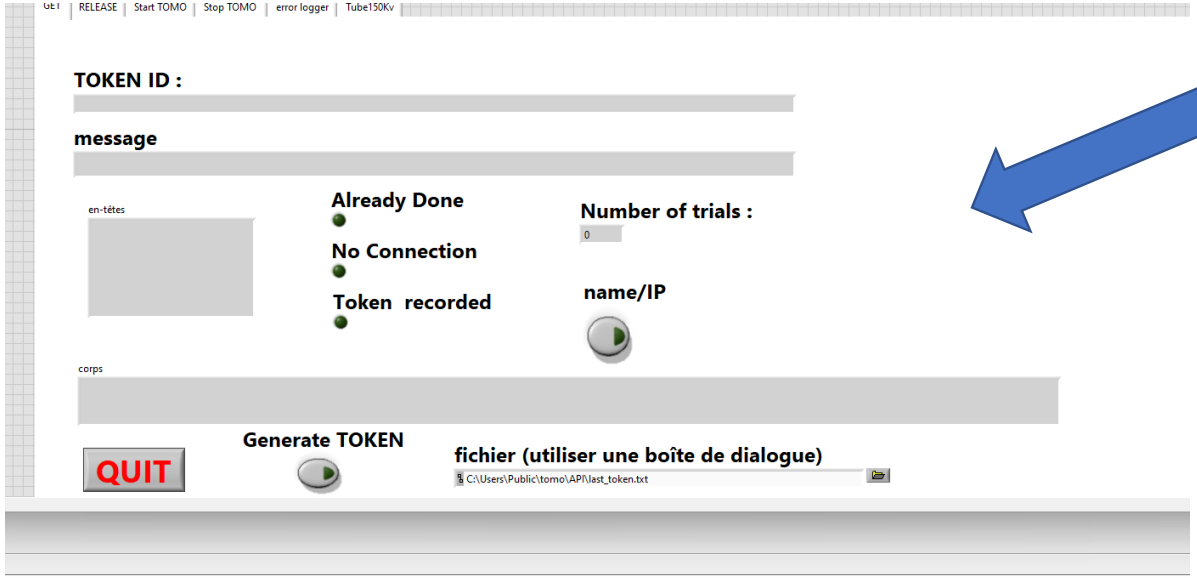

```
{
    "object": "Xray150",
    "operation": "requestObjectInfos",
    "response":
    {
        "message":
        {
            "attributes": [{"name": "connected", "type": "bool"}, {"name": "current", "type": "int"}, {"name": "filter", "type": "QSt
            "commands": [{"name": "requestSpotsize", "parametersType": "double", "returnType": "bool"}, {"name": "setCurrent", "param
            "states": [{"state": "HW_NOT_READY", "value": 1}, {"state": "VOLTAGE", "value": 2}, {"state": "CURRENT", "value": 4}, {"st
        },
        "type": "message"
    }
}
```


2.2. Error log

A global error log is available and permits to trace the event errors on the machine. This log can also be called for each resource as a request (cf. *Requests* section of each resource).

Get the global error log

Get the error log for Xray150 resource



Test commandes principales

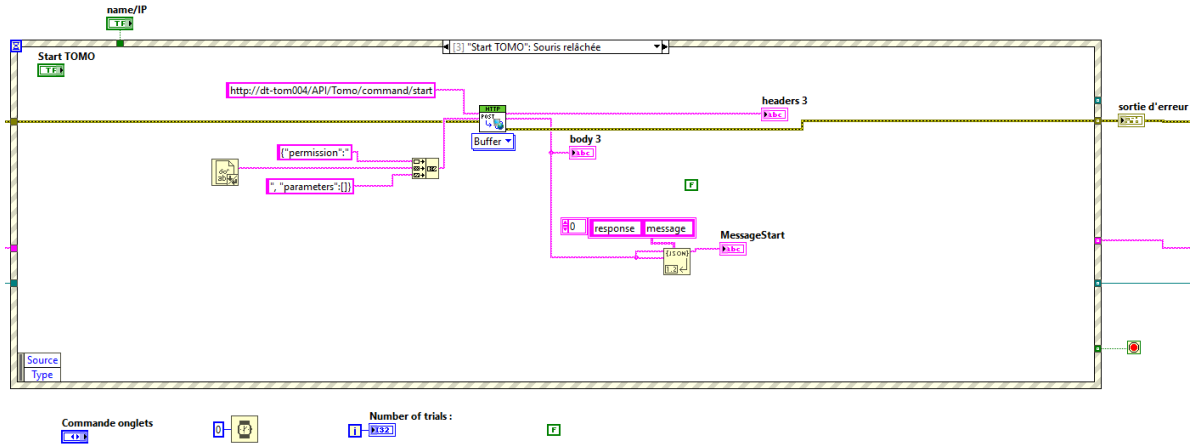
Démarrer Tomographie N°0

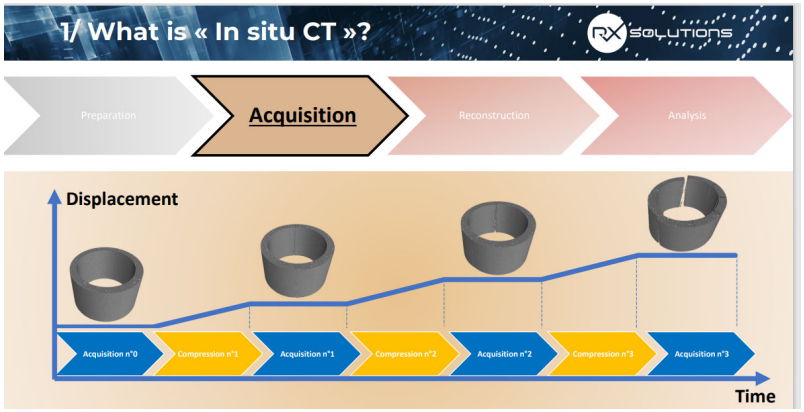
Tester statut Tomo en cours

Si terminée lancer phase de chargement

Tester si Phase de chargement terminée incrementer tomographie suivante

et on cycle juqu'a la tomo finale.





LabVIEW Wiki

Design pattern

A **design pattern**, also know as a software design pattern, is a reusable solution to a software eng templates that you and others can reuse for future projects.

Described below some useful design patterns that a developer can use in their application architect

Basic Design Patterns

State Machine

See more on the State Machine Design Pattern.

Event Handler

See more on the Event Handler Design Pattern.

Master/Slave

See more on the Master/Slave Design Pattern.

Producer/Consumer

See more on the Producer/Consumer Design Pattern.

Intermediate Design Patterns

Queued Message Handler (QMH)

See more on the Queued Message Handler (QMH) Design Pattern.

Queued State Machine (QSM)

See more on the Queued State Machine (QSH) Design Pattern.

Action Engine (AE)

See more on the Action Engine (AE) Design Pattern a.k.a. Functional global variable.

Advanced Design Patterns

Object-Oriented Design Patterns

See more on Object-Oriented Design Patterns.

Actor Oriented Design

See more on the Actor Oriented Design Patterns.

Home
Contents
Featured Content
Categories
A–Z Index
Random Article

interaction

About
Community Portal
Recent Changes
Main Help
How to Edit
Questions
internal

Tools

What links here
Related changes
Special pages
Printable version
Permanent link
Page information