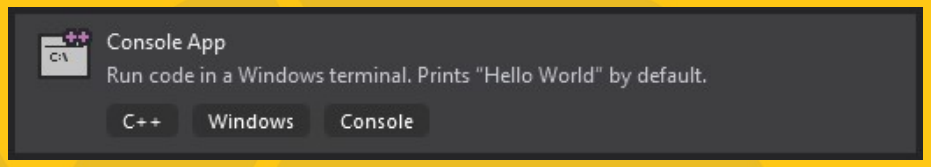
XFS

Prática: desenvolvendo o primeiro APP XFS client

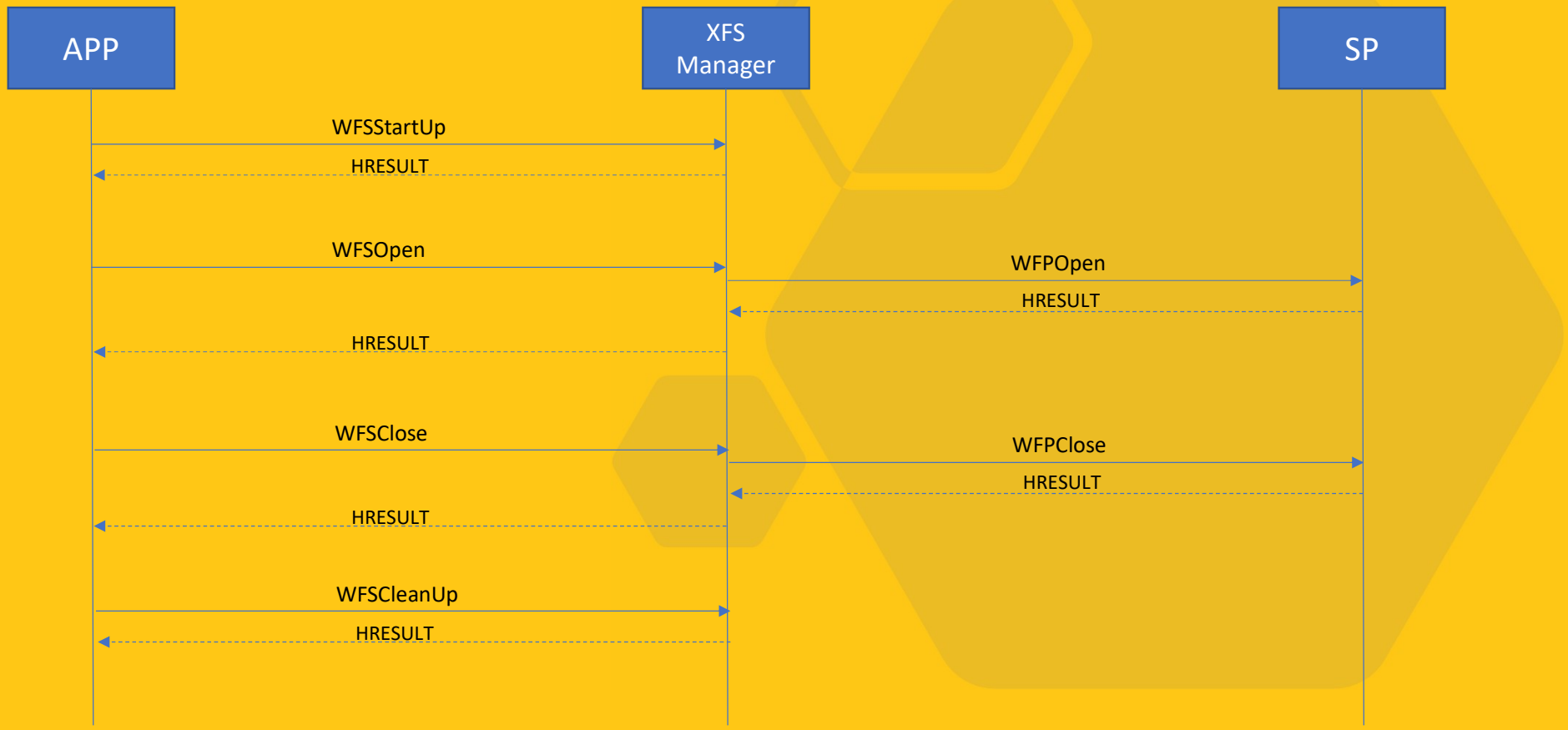


Criar projeto no Visual Studio 2019:

- **Tipo:** Console App
- **Nome do projeto:** xfstest
- **Nome da solução:** xfs
- **Detalhes:**
    1. Ajustar includes e imports conforme slide anterior;
    2. Fazer o include do windows.h
    3. Fazer chamada ao WFSStartUp;
    4. Fazer chamada ao WFSOpen, chamando o seu serviço XFS criado anteriormente;
    5. Fazer chamada ao WFSCleanUp;
    6. Usar uma ferramenta de testes XFS para tentar chamar o seu novo serviço XFS.

XFS

Prática: sequência de chamadas



XFS

Prática: gerenciando versão XFS / SP

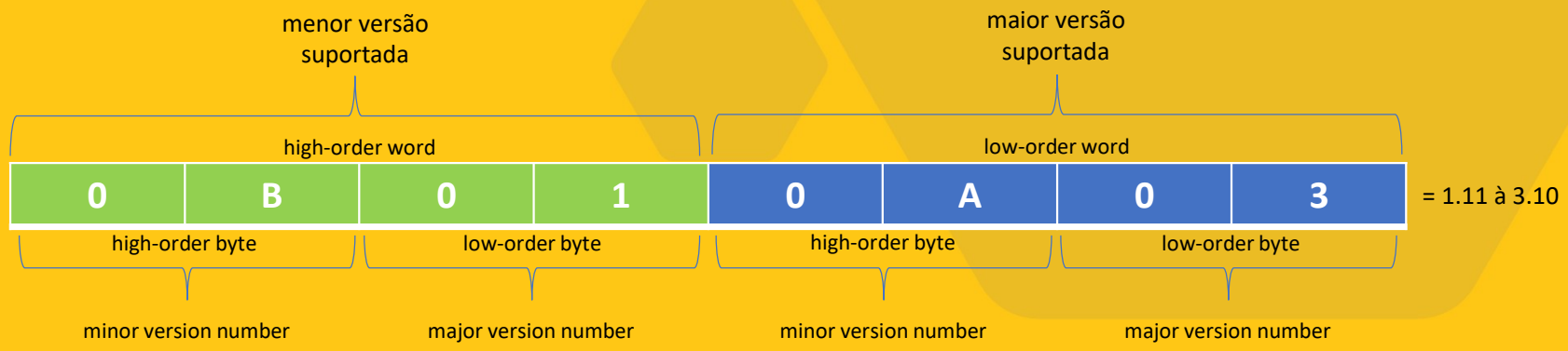
Controle de versão

O XFS Manager oferece um mecanismo de controle de versão, que permite que a APP indique qual é a versão mínima e máxima com a qual ela aceita / pode trabalhar.

Isso impacta no desenvolvimento do SP que precisará também indicar corretamente sua versão e também prover os retornos previstos na especificação para esses casos.

Os comandos WFSStartUp / WFSOpen lidam com essa negociação (campos: dwVersionsRequired / dwSrvcVersionsRequired):

- **WFSStartUp / WFSOpen** : aplicação indica qual versão mínima e máxima que a aplicação consegue executar;
- **WFPOpen**: o XFS Manager indica para o SP a versão mínima e máxima que ele suporta.



tamanho: DWORD / formato: Hexadecimal

XFS

Prática: WFSStartUp

Deve ser sempre o primeiro comando executado pela aplicação. Irá conectar a aplicação ao XFS Manager.

Parâmetros:

dwVersionsRequired: indica o range de versões da XFS API que a aplicação suporta; Notem: raramente há alterações na API (funções WFS). O que normalmente ocorre são atualizações as especificações do SP, introduzindo novos retornos, novas estruturas de dados e etc.

lpWFSVersion: endereço onde o XFS Manager retorna informações sobre versionamento. Criar área local de WFSVersion e passar o endereço;

XFS

Prática: WFSOpen

Abre sessão com o service provider. Ou seja, esse é comando irá refletir na DLL do SP.

Parâmetros:

lpszLogicalName: texto com nome lógico do serviço XFS. Ou seja, o nome configurado na chave LOGICAL_SERVICES.

hApp: Handle da aplicação criado com WFSCreateAppHandle ou WFS_DEFAULT_HAPP;

lpszAppID: texto com ID da aplicação. Pode ser NULL. Quando informado, esse ID é registrado nos traces de eventos e demais mensagens, o que facilita bastante a análise de logs;

dwTraceLevel: Nível de trace a ser setado. Se NULL, todos os níveis de trace são desligados.

dwTimeOut: tempo em milissegundos que a aplicação deseja aguardar até que a operação se complete. Se WFS_INDEFINITE_WAIT, então aguarda até completar, sem considerar timeout.

dwSrvcVersionsRequired: range de versão de SPI que a aplicação suporta.

lpSrvcVersion: ponteiro para área de WFSVersion para receber informações sobre a implementação SPI.

lpSPIVersion: ponteiro para área de WFSVersion onde o SP irá gravar informações de versionamento. Pode ser NULL.

lphService: handle do serviço. Será usado em todas chamadas subsequentes ao SP.

XFS

Prática: WFPOpen

Estabelece conexão entre o service provider e o XFS Manager.

Parâmetros:

hService: handle do serviço. Esse é gerado pelo SP e fica associado a sessão que está sendo aberta.

lpszLogicalName: nome lógico que foi passado pela aplicação. Recebemos isso para obter mais informações no Windows Registry;

hApp: o handle da aplicação para ser associado com a sessão sendo aberta;

lpszAppID: o ID passado pela aplicação. Esse valor pode ser NULL.

dwTraceLevel: nível do trace.

dwTimeOut: tempo em milissegundo que a aplicação vai aguardar até que o SP complete todo o trabalho dessa requisição.

hWnd: endereço da janela (window) criada pelo XFS Manager para que o SP poste mensagens de conclusão dessa requisição.

ReqID: ID dessa requisição. Isso é criado pelo XFS Manager.

hProvider: handle do serviço. O XFS Manager provê isso.

dwSPIVersionsRequired: indica o range de versão do XFS SPI que o XFS Manager suporta.

lpSPIVersion: gravar aqui informação de versionamento do SPI.

dwSrvcVersionsRequired: indica o range de versão do XFS SPI que a aplicação suporta.

lpSrvcVersion: gravar aqui informação de versionamento do SPI.

Campos de versões:

- Quando tiver “SPI” indica que a origem ou destino é o XFS Manager;
- Quando tiver “Srvc” indica que a origem ou destino é a aplicação.
- Normalmente o conteúdo é igual para os campos lpSPIVersion e lpSrvcVersion.

XFS

Prática: adicionar brxutil

- Criar uma pasta common e colocar brxutil.h / brxutil.cpp;
- Desmarcar o uso de Precompiled Headers no SP: **C/C++ - > All Options -> Precompiled Header (Not Using Precompiled Headers)**
- Incluir a pasta “common” no SP e no app: **VC++ Directories - > Include Directories**
- .

XFS

Prática: gerenciamento de memória

Estudar a WFSResult na doc.

O XFS especifica um mecanismo para alocação de liberação de memória. A regra é: o SP aloca memória e a App cliente (ou o XFS Manager) devem liberar essa memória alocada.

O objeto alvo dessa alocação é sempre o WFSRESULT, que contém informações detalhadas sobre a ultima função XFS executada. Esse objeto é trafegado via “completion message”.

Sempre que o SP está pronto para concluir uma chamada WFP, ele deve alocar memória para um objeto WFSRESULT. A alocação é feita com a função **WFMAllocateBuffer**. Se houver necessidade de retornar áreas adicionais junto as propriedades do WFSRESULT, então o SP deve executar o **WFMAllocateMore** quantas vezes for necessário.

A aplicação (ou o XFS Manager) recebe o ponteiro para o objeto WFSRESULT e consegue assim acessar seu conteúdo. Ao completar seu uso, ele deve executar o WFSFreeResult para liberar essa área alocada.

As responsabilidades sobre a liberação da memória são dividas assim:

- **Funções assíncronas da API**: App é responsável por liberar a memória;
- **A funções síncronas WFSExecute, WFSGetInfo e WFSLock**: App é responsável por liberar a memória;
- **Demais funções síncronas da API**: XFS Manager é responsável por liberar a memória;

XFS

Prática: mensagens

O SP deve sempre notificação a conclusão da execução dos comandos WFP, a não ser que o comando tenha terminado com um erro do tipo WFS_ERR_INTERNAL_ERROR.

Essa notificação é feito por meio da publicação de uma mensagem na window que o XFS Manager passa para o SP nas chamadas das funções WFP. As mensagens podem ser do tipo “COMPLETE”, que indica que um comando foi finalizado, mas também pode ser do tipo “EVENT” para indicar que algo inesperado ocorreu. Essas mensagens têm ID definições assim no XFS:

- WFS_OPEN_COMPLETE
- WFS_CLOSE_COMPLETE
- WFS_LOCK_COMPLETE
- WFS_UNLOCK_COMPLETE
- WFS_REGISTER_COMPLETE
- WFS_DEREGISTER_COMPLETE
- WFS_GETINFO_COMPLETE
- WFS_EXECUTE_COMPLETE
- WFS_EXECUTE_EVENT
- WFS_SERVICE_EVENT
- WFS_USER_EVENT
- WFS_SYSTEM_EVENT

Para enviar essas mensagens, a função PostMessage do Win32API deve ser utilizada. O parâmetro WPARAM não é utilizado e o parâmetro LPARAM deve conter o endereço de memória para o objeto WFSRESULT.