// Licensed to the Apache Software Foundation (ASF) under one

// or more contributor license agreements. See the NOTICE file

// distributed with this work for additional information

// regarding copyright ownership. The ASF licenses this file

// to you under the Apache License, Version 2.0 (the

// "License"); you may not use this file except in compliance

// with the License. You may obtain a copy of the License at

//

// http://www.apache.org/licenses/LICENSE-2.0

//

// Unless required by applicable law or agreed to in writing,

// software distributed under the License is distributed on an

// "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY

// KIND, either express or implied. See the License for the

// specific language governing permissions and limitations

// under the License.

#include "arrow/python/parquet\_encryption.h"

#include "parquet/exception.h"

namespace arrow {

namespace py {

namespace parquet {

namespace encryption {

PyKmsClient::PyKmsClient(PyObject\* handler, PyKmsClientVtable vtable)

: handler\_(handler), vtable\_(std::move(vtable)) {

Py\_INCREF(handler);

}

PyKmsClient::~PyKmsClient() {}

std::string PyKmsClient::WrapKey(const std::string& key\_bytes,

const std::string& master\_key\_identifier) {

std::string wrapped;

auto st = SafeCallIntoPython([&]() -> Status {

vtable\_.wrap\_key(handler\_.obj(), key\_bytes, master\_key\_identifier, &wrapped);

return CheckPyError();

});

if (!st.ok()) {

throw ::parquet::ParquetStatusException(st);

}

return wrapped;

}

std::string PyKmsClient::UnwrapKey(const std::string& wrapped\_key,

const std::string& master\_key\_identifier) {

std::string unwrapped;

auto st = SafeCallIntoPython([&]() -> Status {

vtable\_.unwrap\_key(handler\_.obj(), wrapped\_key, master\_key\_identifier, &unwrapped);

return CheckPyError();

});

if (!st.ok()) {

throw ::parquet::ParquetStatusException(st);

}

return unwrapped;

}

PyKmsClientFactory::PyKmsClientFactory(PyObject\* handler, PyKmsClientFactoryVtable vtable)

: handler\_(handler), vtable\_(std::move(vtable)) {

Py\_INCREF(handler);

}

PyKmsClientFactory::~PyKmsClientFactory() {}

std::shared\_ptr<::parquet::encryption::KmsClient> PyKmsClientFactory::CreateKmsClient(

const ::parquet::encryption::KmsConnectionConfig& kms\_connection\_config) {

std::shared\_ptr<::parquet::encryption::KmsClient> kms\_client;

auto st = SafeCallIntoPython([&]() -> Status {

vtable\_.create\_kms\_client(handler\_.obj(), kms\_connection\_config, &kms\_client);

return CheckPyError();

});

if (!st.ok()) {

throw ::parquet::ParquetStatusException(st);

}

return kms\_client;

}

arrow::Result<std::shared\_ptr<::parquet::FileEncryptionProperties>>

PyCryptoFactory::SafeGetFileEncryptionProperties(

const ::parquet::encryption::KmsConnectionConfig& kms\_connection\_config,

const ::parquet::encryption::EncryptionConfiguration& encryption\_config) {

PARQUET\_CATCH\_AND\_RETURN(

this->GetFileEncryptionProperties(kms\_connection\_config, encryption\_config));

}

arrow::Result<std::shared\_ptr<::parquet::FileDecryptionProperties>>

PyCryptoFactory::SafeGetFileDecryptionProperties(

const ::parquet::encryption::KmsConnectionConfig& kms\_connection\_config,

const ::parquet::encryption::DecryptionConfiguration& decryption\_config) {

PARQUET\_CATCH\_AND\_RETURN(

this->GetFileDecryptionProperties(kms\_connection\_config, decryption\_config));

}

} // namespace encryption

} // namespace parquet

} // namespace py

} // namespace arrow